CENTRE DE RECHERCHE SUR LES TRANSPORTS
(C.R.T.)
CENTRE FOR RESEARCH ON TRANSPORTATION

UNIVERSITÉ DE MONTRÉAL
ÉCOLE DES HAUTES ÉTUDES COMMERCIALES
ÉCOLE POLYTECHNIQUE

MÉMOIRE D'ÉTUDIANT

CENTRE DE RECHERCHE SUR LES TRANSPORTS
UNIVERSITÉ DE MONTRÉAL
C.P. 6128, SUCCURSALE A.
MONTRÉAL
CANADA H3C 3J7

TÉLÉPHONE: (514) 343-7575
TÉLEX: 05-24146 BIBPOLYTEC
EMAIL: crt@crt.umontreal.ca
TÉLÉCOPIEUR (FAX): (514) 343-7121

90-06429/

CRT-699

CALCUL SIMULTANÉ DE LA MATRICE DE DEMANDE ET DES FLOTS OPTIMAUX DE MARCHANDISES SUR UN RÉSEAU MULTIMODAL

par

Christine Gédéon*

* Mémoire de maîtrise, département d'informatique et de recherche opérationnelle, Université de Montréal, dirigé par Michael Florian.

Centre de recherche sur les transports - Publication #699

Juin 1990

Dépôt légal - 2e trimestre 1990

Sommaire

Cette étude se propose de développer un algorithme pour résoudre le problème du calcul simultané de la matrice de demande et des flots optimaux de marchandises sur un réseau multimodal, sachant que les flots sont contrôlés par l'usager, et que seuls sont connus les totaux marginaux de la matrice, le réseau et sa structure de coûts, celle-ci pouvant varier d'un produit à l'autre.

Considérant un modèle normatif et la recherche de l'optimum du système, principales contributions de ce mémoire, nous avons adapté la méthode d'approximation linéaire de Frank et Wolfe en supposant que les fonctions de coût étaient convexes, continues et partout différentiables sur le domaine réalisable, et que le réseau était connexe.

Deux niveaux de difficulté se dégagent de la formulation du problème, selon que l'on a un ou plusieurs produits à transporter.

Pour un seul produit, il s'agit de résoudre l'approximation linéaire de l'objectif qui revient à un problème de transport entre les origines et les destinations avec pour coûts les coûts marginaux de la solution courante. On détermine alors, la direction de diminution, puis le pas optimal à faire pour calculer la nouvelle solution.

Excepté quelques cas particuliers, le problème multiproduit nécessite des manipulations supplémentaires pour la résolution de l'approximation linéaire, à cause de la présence de plusieurs matrices par produits. Au terme de

celles-ci, on retrouve un problème de transport classique. Dans tous les cas, cependant, les coûts unitaires de ce dernier problème sont plus complexes.

L'algorithme adapté au premier cas a été implanté. Des tests, consistant en son application et l'application d'un algorithme d'affectation avec matrice fixe, ont été effectués sur les données réelles du réseau d'importation de charbon en Finlande. Ils démontrent qu'en cas de structure de coûts linéaire ou légèrement non-linéaire le nouvel algorithme est très intéressant. Les résultats, en revanche, ne sont guère encourageants quand la structure est fortement non-linéaire.

Outre tester le comportement de la version multiproduit, des études complémentaires pourraient envisager une méthode alternative: résoudre un problème de flots à coûts minimal, puis décomposer les flots obtenus en flots de chemin pour trouver une matrice correspondante. Les résultats obtenus permettront de mieux apprécier l'intérêt de l'algorithme présenté.

Mots clefs: modèles d'optimisation sur réseau, multiflot, problèmes de distribution.

Table des matières

Sommaire i

Table des matières iii

Liste des tableaux vi

Liste des figures vii

Introduction 1

I Le modèle mathématique 4

1. Terminologie 4
2. Notation 7
3. Hypothèses de base sur l'objectif et le réseau 9
4. Le modèle de base: un produit, un mode 11
5. Généralisation: un produit, plusieurs modes 14
6. Généralisation: plusieurs produits et modes 15

II Approche algorithmique 17

1. L'algorithme de Frank et Wolfe 17
2. Le sous-problème linéaire 19
3. La direction de diminution 22

4. Le pas optimal 23
5. La mise à jour des variables 23
6. Le critère d'arrêt 24
7. L'initialisation 26
7.1. Génération à partir d'un flot nul 27
7.2. Méthode de balancement à deux dimensions 28

III Généralisation de l'approche **31**

IV Cas particuliers **39**
1. Réseau à coûts unitaires constants 39
2. Un seul regroupement de modes par produit 41
3. Information supplémentaire sur les $w^{m(p)}$ 43

V Implantation pratique **45**
1. Outils et choix d'algorithme 45
1.1. Outils disponibles 45
1.2. Problématique de choix 46
1.3. Algorithme implanté 47
2. Structure et organisation 47
2.1. Structure du programme 47
2.2. Organisation générale 48
2.3. Dialogues et rapports 49
2.4. Environnement de travail 50
3. Détails pratiques de l'implantation 50
3.1. Sous-programme 1 : Création d'une matrice initiale 50
3.2. Sous-programme 2 : Exécution de l'algorithme 51
3.3. Limitations du module expérimental 56

VI Résultats numériques 58

1. Les caractéristiques du réseau finlandais 58

1.1. Le réseau de test 58

1.2. Les modes 60

2. Les scénarios des tests 64

3. Résultats des tests 68

3.1. Groupe I: fonctions linéaires 70

3.2. Groupe II: fonctions linéaires et faiblement non-linéaires ... 70

3.3. Groupe III: fonctions linéaires et fortement non-linéaires ... 70

4. Commentaires 72

Conclusion 74

Bibliographie 76

Bibliographie thématique 79

A Liste des programmes i

1. Liste du contrôleur du module s551 i

2. Liste du sous-programme d'initialisation sm1 ii

3. Liste du sous-programme d'exécution sm2 iv

4. Liste de l'interface avec RNET xii

B Introduction à STAN xvi

1. Environnement de la banque xvi

2. Les modules et leurs fonctions xvii

3. Bibliographie xx

Remerciements xxi

Liste des tableaux

I Symboles et notations 7–8

II Caractéristiques physiques du réseau finlandais 59

III Réseau navigable finlandais 60

IV Description des modes du réseau finlandais 63

V Description des divers tests numériques 67

VI Résultats par test 68

VII Comparaison des résultats des tests 72

Liste des figures

1 Représentation d'un réseau multimodal sous forme unimodale ... 10

2 Fonction intégrant la capacité 11

3 Le *gap* d'une fonction à une seule inconnue 26

4 Réseau figurant le problème (54) – (57) 35

5 Organisation du module **5.51** 48

6 Environnements de travail 51

7 Flot d'information dans le programme **sm2** 53

8 Le réseau de transport finlandais 61

9 Exemples de fonctions de coûts 66

10 Comportement du *gap* en fonction du nombre d'itérations 69

11 Comportement de l'objectif en fonction du nombre d'itérations .. 71

Introduction

Nous savons quelles sont les quantités de charbon disponibles pour l'importation en divers endroits outremer. Nous savons quelles sont celles dont nous avons besoin dans les diverses villes du pays. Nous désirons savoir pour chacune de ces villes comment importer le charbon. Voilà la question que nous a posé le ministère des transports de Finlande.

Derrière ce problème apparemment spécifique se cache un autre beaucoup plus général, que l'on pourrait formuler ainsi:

- Sachant quelles quantités de marchandises sont disponibles aux sources (ou origines) d'un réseau,
- sachant de quelles quantités nous avons besoin à chaque destination de ce même réseau,
- nous voulons savoir quelle proportion de la quantité disponible à telle origine envoyer à quelle destination, et comment (c'est-à-dire par quel cheminement) de façon à encourir le moindre coût.

En résumé, nous voulons trouver, et la matrice origine/destination (matrice O/D), et les flots optimaux sur le réseau.

Il y a donc deux composantes à notre problème: un problème de distribution pour trouver la matrice O/D, et un problème d'affectation pour trouver les flots optimaux.

Dans un contexte où les flots représentent des déplacements de personnes, la principale difficulté se trouve au niveau du problème de distribution. En effet, la matrice de demande est souvent variable, et de plus, nous n'avons aucun contrôle sur elle. Par contre, il est très important que la solution du problème de distribution donne des flots réalistes après son affectation (c'est-à-dire qu'ils reproduisent les flots observés). A ce chapitre, le problème est souvent résolu en pratique par des spécialisations de la méthode de balancement de Bregman (voir [16]), mais il existe aussi une adaptation de la méthode de Newton proposée par Robillard et Stewart [22]. Andersson [2] a fait une étude comparative d'une des méthodes de balancement, celle dite de Furness [13], et de la méthode de Newton. Le lecteur y trouvera une analyse complète de leur convergences respectives. Il trouvera dans [21] une revue assez complète des modèles et algorithmes utilisés dans ces situations.

Dans le contexte du transport de marchandises, le nôtre, ce problème, ne se pose pas. En effet, le gestionnaire jouit généralement du contrôle total sur la distribution des produits: la contrainte de vraisemblance des flots calculés est peu, ou pas importante. Par contre, celle de moindre coût n'en a qu'une priorité accrue. Le problème de transport classique et ses algorithmes de résolution bien connus permettent de résoudre la question de distribution dans ce cas.

Certes, le problème de distribution est un aspect important de la question; et l'on serait tenté de considérer qu'une fois sa résolution complétée, il suffit, pour avoir la solution recherchée, d'affecter la matrice ainsi trouvée au réseau, c'est-à-dire, de trouver les itinéraires qui minimisent les coûts totaux de déplacement. C'est compter sans la congestion! En effet, un changement dans la distribution du flot sur un réseau congestionné, peut entraîner une grosse variation de coût, de sorte que les coûts des flots induits par la

matrice résultant du problème de distribution ne reflètent souvent pas ceux avec lesquels cette même matrice a été calculée. La solution intuitive à ce problème, qui consisterait à répéter successivement distribution, puis affectation ne converge malheureusement vers rien [11].

Comment alors trouver une matrice qui reflète les coûts optimaux absolus sur le réseau, sachant par ailleurs que cette matrice ne saurait être unique? La solution à cette question est sans doute dans les modèles combinés de distribution-affectation. Le premier à avoir énoncé un tel modèle et une méthode de résolution convergente est Tomlin [23]. Il proposait de résoudre la question par la technique de décomposition et de génération de colonnes de Dantzig-Wolfe. On trouvera aussi dans [7], [11], [10] et [6] d'autres approches plus récentes de la question. Cependant, elles ont toutes en commun (sauf pour [23]) la caractéristique d'utiliser le modèle descriptif du comportement des flots qui cherche à minimiser le coût moyen sur les arcs du réseau. Elles ont également toutes en commun de supposer que l'on tend vers un équilibre qui traduit le principe égoïste de Wardrop [24]. Notre contribution sera de considérer un modèle normatif cherchant à minimiser les coûts totaux de transport, et non ceux de chaque particule de flot individuelle.

Pour ce faire, notre démarche sera la suivante: au chapitre I nous exposerons le modèle proposé en divers degrés de généralité, puis nous développerons au chapitre II une approche de résolution basée sur la méthode d'approximation linéaire [12]. Au chapitre III nous généraliserons cette approche, et au chapitre IV nous analyserons quelques cas particuliers. Nous décrirons au chapitre V une implantation informatique de notre méthode. Nous passerons en revue les tests et résultats numériques obtenus grâce à cette implantation au chapitre VI, avant de tirer quelques conclusions de cette étude.

Chapitre I Le modèle mathématique

Dans le présent chapitre, nous établirons la formulation mathématique du problème dans le cas le plus simple où un seul produit est transporté grâce à un mode de transport unique. Nous aborderons ensuite la généralisation aux problèmes plus courants au niveau où de telles études interviennent. Cette généralisation se fera en deux étapes. Dans une première, nous verrons comment formuler le problème de distribution et d'affectation d'un produit unique par un éventail de modes de transport possibles; dans une seconde nous en établirons la formulation dans le cas de plusieurs produits ayant chacun son éventail de modes, ou de regroupements de modes, de transport possibles.

Mais auparavant nous allons développer le formalisme nécessaire à notre démarche, à commencer par la terminologie. La notation et les hypothèses de base qui suivront demeureront valables pour tout le reste de cette étude.

1. Terminologie

Les marchandises, denrées, biens d'équipement, matières premières, *etc* ..., faisant l'objet du problème de distribution seront appelés **produits**.

Le graphe orienté représentant mathématiquement le réseau de transport physique proposé sera appelé **réseau**. Par analogie à la terminologie de la théorie des graphes, nous parlerons de **sous-réseau** pour sous-graphe, de réseau connexe pour graphe connexe, de **nœuds** et d'**arcs** du réseau.

Un réseau peut n'offrir aucun choix en ce qui concerne le mode de transport. En pareil cas, nous parlerons de **réseau unimodal**; sinon, nous parlerons de **réseau multimodal**. Pour analyser de façon plus détaillée les flots, les coûts sur le réseau et l'impact d'un changement de ces derniers dans un réseau multimodal, il arrive que l'on considère séparément divers sous-ensembles, non nécessairement disjoints, des modes accessibles à un produit donné, et que l'on attribue à chacun d'entre eux une matrice de demande propre. Nous appellerons **regroupement de modes** chacun de ces sous-ensembles.

Nous distinguerons trois catégories de nœuds:

- les **nœuds simples**, qui correspondent à des intersections, des changements d'orientation ou de tout autre caractéristique similaire dans le réseau physique,
- les **nœuds de transfert**, correspondant à des points du réseau physique où sont disponibles les équipements nécessaires au transbordement du produit des véhicules d'un mode à ceux d'un autre,
- les **centroïdes**, nœuds spéciaux, proportionnellement peu nombreux en général, qui représentent les régions de départ ou d'aboutissement du trafic des produits sur le réseau; ils sont également appelés **zones**.

  On distingue les **centroïdes d'offre**, également appelés **origines**, où est disponible le produit à distribuer, et les **centroïdes de demande**, également appelés **destinations**, où ce produit est requis.

Les arcs, quant à eux, sont de deux sortes:

- Les **liens**: ils représentent le cheminement physique du produit à transporter entre deux nœuds simples du réseau, ou entre un centroïde et un nœud simple.

  Ils sont caractérisés par un certain nombre d'attributs, principalement la fonction de coût de transport, la capacité maximale de transport, la longueur, le volume de flot traversant l'arc, le mode de transport qui lui est associé, le nœud de départ et celui d'arrivée.

- Les **transferts**: ils représentent le transbordement du produit en un nœud donné, des véhicules d'un mode à ceux d'un autre.

  Tout comme les liens, ils ont divers attributs, dont, une fonction de coût, une capacité, le volume du flot de produit transbordé, le mode de provenance, celui de destination et le nœud où a lieu le transfert.

Un **chemin** reliant une origine donnée à une destination donnée est une suite orientée connexe d'arcs telle que l'on puisse partir de l'origine et arriver à la destination en empruntant ces arcs dans le sens de leur orientation.

Plus en rapport avec le problème de distribution, quand nous parlerons de la **demande** d'un produit, il s'agira, selon le contexte:

- de la caractéristique d'une destination donnée, à savoir, la quantité de ce produit qui y est requise,
- de la quantité totale de ce produit, demandée dans l'ensemble des destinations du réseau,
- d'un raccourci pour "matrice de demande" (voir ci-après).

Par analogie au premier sens de la demande, nous désignerons par **l'offre** d'un produit à une origine donnée, la quantité de produit qui y est disponible.

Enfin, nous appellerons **matrice de demande** (ou **matrice O/D**) d'un

produit, une matrice à deux dimensions, dont les indices de la première dimension représentent les origines, et ceux de la deuxième représentent les destinations, et dont chaque élément correspond à la quantité totale de produit circulant spécifiquement entre l'origine et la destination que représente la paire d'indices considérée.

2. Notation

Bien que cela puisse paraître peu — ou pas — naturel, puisque notre étude en a pris la démarche inverse, nous allons présenter la notation dans le contexte du problème le plus général où plusieurs modes et plusieurs produits entrent en jeu. Cette façon de faire évitera d'inutiles répétitions, puisque la notation la plus générale se distingue des autres par la simple inclusion d'indices additionnels. Il sera très facile de se ramener à un cas plus particulier en laissant tomber l'indice devenu inutile pour l'application considérée, ou encore en considérant que l'ensemble correspondant n'a qu'un seul élément.

Le tableau ci-dessous résume toute la notation:

Symbole	Concept
$\mathcal{N}$	L'ensemble des nœuds du réseau.
$\mathcal{A}$	L'ensemble des arcs reliant les nœuds du réseau.
$\mathcal{O}$	L'ensemble des centroïdes d'offre.
$\mathcal{D}$	L'ensemble des centroïdes de demande.
$\mathcal{P}$	L'ensemble des produits à distribuer.
$\mathcal{M}$	L'ensemble des modes disponibles.

Tableau I: Symboles et notations.

Symbole	Concept
a	Un arc déterminé.
o	Une origine déterminée.
d	Une destination déterminée.
p	Un produit déterminé.
m	Un mode déterminé.
O_o^p	L'offre du produit p à l'origine o.
D_d^p	La demande du produit p à la destination d.
$\mathcal{M}_p$	L'ensemble des modes accessibles au produit p.
$\mathcal{M}(p)$	L'ensemble des regroupements de modes pour le produit p.
$m(p)$	Un regroupement de modes du produit p et le sous-réseau associé.
v_a^p ou y_a^p	Le volume du produit p transporté dans l'arc a.
c_a^p	La capacité de l'arc a pour ce qui est du produit p.
$s_a^p(\cdot)$	La fonction de coût associée à l'arc a pour le transport du produit p.
$K_{od}^{m(p)}$	L'ensemble des chemins reliant la paire de centroïdes (o,d) dans le sous-réseau $m(p)$ du produit p.
k	Un chemin déterminé.
h_k ou z_k	Le volume total du flot sur le chemin k.
$g_{od}^{m(p)}$	La quantité de produit p envoyée de o à d par les chemins du sous-réseau $m(p)$.
$w_{od}^{m(p)}$	*idem.*

Tableau I: (suite)

Remarquons que, lorsque nous référons à l'arc a, nous faisons implicitement référence, si cet arc est un lien, au mode qu'il représente et aux nœuds qu'il relie. De même, s'il s'agit d'un transfert, la référence est implicite au nœud où il a lieu, au mode de provenance, et à celui de destination. Il n'est donc pas nécessaire d'associer à ces divers attributs de notation particulière, d'autant plus qu'ils n'entrent pas en jeu au niveau de la modélisation et de la programmation mathématique.

Notons enfin que nous serons amenés, à certains moments, à introduire des notations supplémentaires pour alléger des expressions devenues trop lourdes à manipuler. Celles-ci ne seront présentées qu'au fur et à mesure des besoins, puisqu'elles ne sont pas directement liées aux concepts de base du problème.

3. Hypothèses de base sur l'objectif et le réseau

Avant de poursuivre notre exposé, posons une fois pour toutes les hypothèses que nous supposerons toujours vérifiées dans la suite de ce rapport.

La fonction objectif du problème à résoudre, c'est-à-dire, en fait, les fonctions de coût de chaque arc prises individuellement, est supposée convexe, continue et partout différentiable dans le domaine de définition, c'est-à-dire dans le domaine des flots réalisables. Cette hypothèse est essentielle pour l'application de l'algorithme de résolution proposé. De plus, les coûts sont supposés séparables, c'est-à-dire que le coût $s_a^p(\cdot)$ dépend uniquement du flot sur l'arc a.

Le réseau de transport est constitué de $|\mathcal{M}|$ sous-réseaux connexes, deux à deux disjoints et ne contenant chacun que des liens. La connexité du réseau global est assurée par les transferts qui relient entre eux les nœuds

"homologues" de ces divers sous-réseaux (voir la figure 1 page suivante). Cette hypothèse nous permet de traiter les transferts de la même manière que les liens. Elle justifie également l'emploi d'une notation unique pour représenter les uns et les autres.

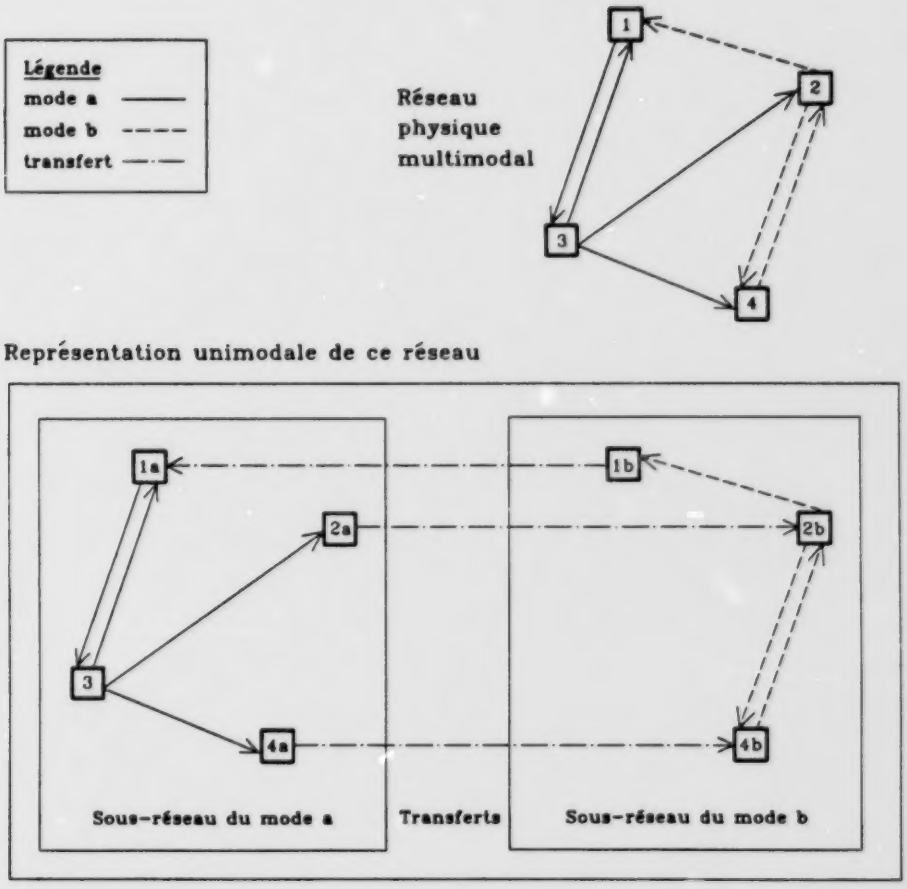


Figure 1: Représentation d'un réseau multimodal sous forme unimodale.

D'autre part, nous supposerons que les fonctions de coût de tous les arcs ayant des capacités de flot restreintes incluent ces contraintes sous forme de termes de pénalité qui en assurent le respect implicite. (voir la figure 2 page suivante).

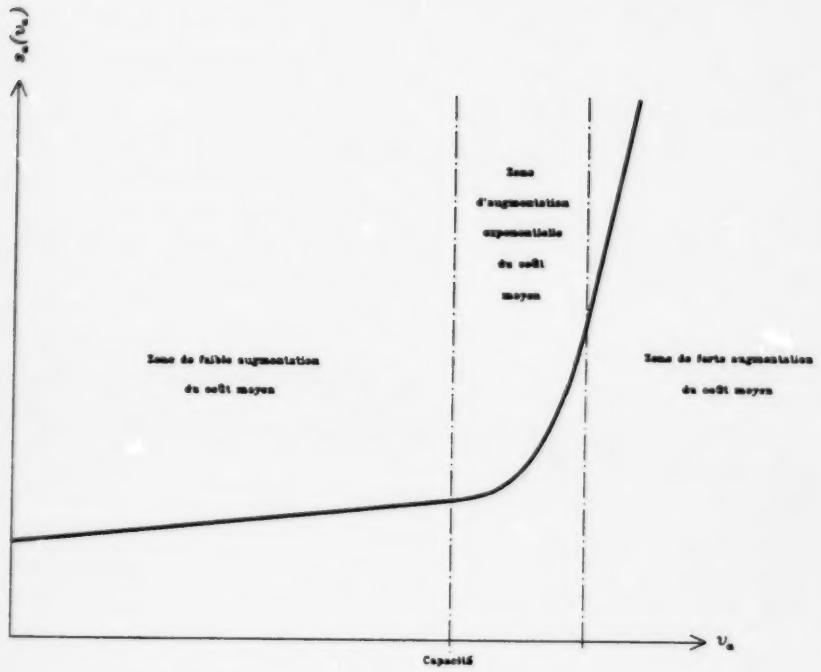


Figure 2: Fonction intégrant la capacité.

4. Le modèle de base: un produit, un mode

Un réseau de transport où le choix du mode de transport serait inexitant pour un produit donné est un cas vraiment rare, qui ne se présente que dans des circonstances particulières. C'est celui, par exemple, des denrées périssables, pour lesquelles seul un moyen de transport extrêmement rapide est acceptable, ce qui élimine *de facto* la considération de tout autre. Un autre cas unimodal possible serait celui du transport d'une matière dangereuse, où les exigences de sécurité sont souvent telles que la liberté de choix parmi les modes disponibles devient presque inexistante.

La formulation A ci-dessous donne un modèle mathématique du problème dans de telles situations.

$$\min_{(v,g)} \sum_{a \in \mathcal{A}} v_a \cdot s_a(v_a) \tag{1}$$

Sujet à:

$$-\sum_{k \in K_{od}} h_k + g_{od} = 0, \qquad \forall o \in \mathcal{O}, \forall d \in \mathcal{D} \tag{2}$$

$$\sum_{d \in \mathcal{D}} g_{od} = O_o, \qquad \forall o \in \mathcal{O} \tag{3}$$

$$\sum_{o \in \mathcal{O}} g_{od} = D_d, \qquad \forall d \in \mathcal{D} \tag{4}$$

$$h_k \geq 0, \qquad \forall k \in K_{od}, \forall o \in \mathcal{O}, \forall d \in \mathcal{D} \tag{5}$$

$$g_{od} \geq 0, \qquad \forall o \in \mathcal{O}, \forall d \in \mathcal{D} \tag{6}$$

$$v_a = \sum_{o \in \mathcal{O}} \sum_{d \in \mathcal{D}} \sum_{k \in K_{od}} \delta_{ak} \cdot h_k, \qquad \forall a \in \mathcal{A} \tag{7}$$

Avec: $\delta_{ak} = \begin{cases} 1 & \text{Si l'arc } a \text{ appartient au chemin } k \\ 0 & \text{Sinon} \end{cases}$

Formulation A: Problème de base, réseau unimodal.

Notons au passage la signification des diverses contraintes:

- L'équation (2) modélise la nécessité de conservation du flot à chaque nœud du réseau.
- L'équation (3) fait de même pour la satisfaction stricte de la demande.
- L'équation (4) garantit la disponibilité stricte des quantités demandées.

- L'équation (5) garantit la faisabilité de la solution (c'est-à-dire la non négativité des flots).
- L'équation (6) est redondante, et garantit également la non-négativité des flots. Elle découle de la combinaison des équations (2) et (5) et ne se trouve ici que par souci de symétrie.
- L'équation (7) ne constitue pas réellement une contrainte. C'est une définition qui nous permet de passer aisément du problème exprimé en fonction du flot sur les arcs du réseau, à son expression en termes du flot sur les chemins entre les paires (o, d). Elle est la conséquence même de la définition d'un chemin.

Avant de passer à la généralisation du modèle, remarquons, premièrement, que seul l'objectif du problème (1) - (7) est — ou, plus exactement, peut être — non linéaire, alors que le domaine réalisable se trouve défini par des contraintes linéaires.

Il est aussi intéressant de remarquer que si, dans la formulation A nous modifions l'objectif comme suit:

$$\min_{(v,g)} \sum_{a \in \mathcal{A}} v_a \cdot s_a(v_a) + \frac{1}{\theta} \cdot \sum_{o \in \mathcal{O}} \sum_{d \in \mathcal{D}} g_{od} \cdot \ln g_{od} \tag{8}$$

où θ est un paramètre libre à calibrer, nous retrouvons un modèle connu: le modèle combiné de distribution-affectation dont la version descriptive ci-dessous a déjà été abondamment étudiée (voir notamment [7], [10] et [11]).

$$\min_{(v,g)} \sum_{a \in \mathcal{A}} \int_0^{v_a} s_a(u) \cdot du + \frac{1}{\theta} \cdot \sum_{o \in \mathcal{O}} \sum_{d \in \mathcal{D}} g_{od} \cdot \ln g_{od} \tag{8'}$$

5. Généralisation: un produit, plusieurs modes

$$\min_{(v,g)} \sum_{a \in \mathcal{A}} v_a \cdot s_a(v_a) \tag{9}$$

Sujet à:

$$-\sum_{k \in K_{od}} h_k + g_{od} = 0, \qquad \forall o \in \mathcal{O}, \forall d \in \mathcal{D} \tag{10}$$

$$\sum_{d \in \mathcal{D}} g_{od} = O_o, \qquad \forall o \in \mathcal{O} \tag{11}$$

$$\sum_{o \in \mathcal{O}} g_{od} = D_d, \qquad \forall d \in \mathcal{D} \tag{12}$$

$$h_k \geq 0, \qquad \forall k \in K_{od}, \forall o \in \mathcal{O}, \forall d \in \mathcal{D} \tag{13}$$

$$g_{od} \geq 0, \qquad \forall o \in \mathcal{O}, \forall d \in \mathcal{D} \tag{14}$$

$$v_a = \sum_{o \in \mathcal{O}} \sum_{d \in \mathcal{D}} \sum_{k \in K_{od}} \delta_{ak} \cdot h_k, \qquad \forall a \in \mathcal{A} \tag{15}$$

Avec: $\delta_{ak} = \begin{cases} 1 & \text{Si l'arc } a \text{ appartient au chemin } k \\ 0 & \text{Sinon} \end{cases}$

Formulation B: Généralisation à plusieurs modes

Comme déjà dit précédemment, les cas unimodaux sont loin de représenter la majorité des cas réels, car le réseau de transport offre en général le choix entre plusieurs modes pour effectuer la tâche. Le fait que ces modes soient plus ou moins pratiques relève du domaine de la calibration des fonctions de coût et non de la modélisation mathématique, aussi, comme le montre la formulation B ci-dessus, la représentation mathématique du problème dans le cas multimodal ne diffère-t-elle pas de celle déjà développée. En effet, les hypothèses de travail, présentées à la section 3 de ce même chapitre, nous permettent d'introduire implicitement la multimodalité, aussi bien dans la définition de l'ensemble $\mathcal{A}$ des arcs, que dans celle des chemins k.

6. Généralisation: plusieurs produits et modes

$$\min_{(v,g)} \sum_{p\in\mathcal{P}} \sum_{a\in\mathcal{A}} v_a^p \cdot s_a^p(v_a) \tag{16}$$

Sujet à:

$$-\sum_{k\in K_{od}^{m(p)}} h_k + g_{od}^{m(p)} = 0, \qquad \forall o\in\mathcal{O}, \forall d\in\mathcal{D}, \forall m(p)\in\mathcal{M}(p)^{[1]} \tag{17}$$

$$\sum_{m(p)\in\mathcal{M}(p)} \sum_{d\in\mathcal{D}} g_{od}^{m(p)} = O_o^p, \qquad \forall o\in\mathcal{O}^{[1]} \tag{18}$$

$$\sum_{m(p)\in\mathcal{M}(p)} \sum_{o\in\mathcal{O}} g_{od}^{m(p)} = D_d^p, \qquad \forall d\in\mathcal{D}^{[1]} \tag{19}$$

$$h_k \geq 0, \qquad \forall k\in K_{od}^{m(p)}, \qquad \forall o\in\mathcal{O}, \forall d\in\mathcal{D}, \forall m(p)\in\mathcal{M}(p)^{[1]} \tag{20}$$

$$g_{od}^{m(p)} \geq 0, \qquad \forall o\in\mathcal{O}, \forall d\in\mathcal{D}, \forall m(p)\in\mathcal{M}(p)^{[1]} \tag{21}$$

$$v_a = \sum_{p\in\mathcal{P}} v_a^p, \qquad \forall a\in\mathcal{A} \tag{22}$$

$$v_a^p = \sum_{o\in\mathcal{O}} \sum_{d\in\mathcal{D}} \sum_{m(p)\in\mathcal{M}(p)} \sum_{k\in K_{od}^{m(p)}} \delta_{ak} \cdot h_k, \quad \forall a\in\mathcal{A}^{[1]} \tag{23}$$

Avec : $\delta_{ak} = \begin{cases} 1 & \text{Si l'arc } a \text{ appartient au chemin } k \\ 0 & \text{Sinon} \end{cases}$

[1] $\forall p\in\mathcal{P}$

Formulation C: Généralisation à plusieurs produits

Les organismes, privés ou publics, qui ont à faire avec des problèmes de distribution ne se contentent généralement pas du transport d'un seul produit. Il est donc normal d'envisager aussi un modèle mathématique qui permettrait de traiter ce type de cas, qui sont les plus généraux. Pour des raisons aisément imaginables, comme par exemple l'encombrement physique du produit, sa périssabilité, son caractère dangereux, les traitements particuliers qu'il nécessite, ... *etc.*, le réseau de transport n'est alors plus unique, dans la

mesure où un produit donné p peut n'avoir accès qu'à un sous-ensemble des modes possibles correspondant à un sous-réseau. En outre, les sous-réseaux ainsi définis pour les divers produits sont en général différents et se recouvrent partiellement. Enfin, il arrive que certains modes de transport aient, par exemple, une direction unique, ou une politique de prix semblable, ou encore des installations de livraison conjointes, de sorte qu'en les regroupant séparément et en leur assignant une matrice de demande particulière, l'on puisse analyser l'impact de tout changement pratique (équipement, nombre de personnel, politique salariale, *etc* ...,) qui se réflèterait sur la structure de coûts du réseau de distribution. C'est cette dernière possibilité qui rajoute un degré non négligeable de complexité au problème proposé. Nous verrons en effet ultérieurement, que lorsqu'elle est inexistante ou qu'elle s'accompagne de données complémentaires adéquates, le problème s'en trouve considérablement simplifié.

La formulation C donnée plus haut exprime le problème dans sa version la plus complexe et sert donc de conclusion aux généralisations successives que nous avons menées.

Les diverses formulations développées ci-dessus vont maintenant servir de point de départ pour l'approche algorithmique du prochain chapitre.

Chapitre II Approche algorithmique

Dans les paragraphes qui suivent, nous développerons une adaptation, au problème de base présenté au chapitre précédent (formulation A), de l'algorithme d'approximation linéaire proposé par Frank et Wolfe [12]. Nous examinerons plus particulièrement la définition du problème approximé, l'étude du critère d'optimalité à choisir pour une implantation informatique de l'algorithme, et le problème de la génération automatique d'une matrice de demande initiale.

1. L'algorithme de Frank et Wolfe

L'algorithme de Frank et Wolfe procède par minimisations successives de l'objectif linéarisé du problème. Cette façon de procéder est d'autant plus attrayante que les contraintes du problème sont elles-même linéaires. De plus, la méthode ne faisant appel qu'au gradient, son implantation informatique est réaliste, même pour de gros réseaux. Enfin, en sus de sa simplicité conceptuelle et de la facilité de son implantation sur ordinateur, cet algorithme à l'avantage de converger très rapidement au cours des premières itérations. Certes, son taux de convergence devient sous-linéaire au voisinage de l'optimum, mais il ne s'agit cependant pas d'un désavantage handicapant:

le type particulier de problèmes que nous envisageons dans cette étude requiert en effet des décisions à un niveau stratégique (par exemple au niveau d'un gouvernement, ou encore celui de la gestion d'une entreprise multinationale, ... *etc.*), plutôt qu'opérationnel; par conséquent, le but recherché n'est pas tant la précision des résultats que l'estimation globale d'une situation donnée.

Soit à résoudre le problème suivant :

$$\min\{\mathcal{F}(x) : x \in D\}$$

où $\mathcal{F}(x)$ est une fonction convexe et D un polytope défini par les contraintes linéaires du problème. L'algorithme de résolution, tel que présenté par M. Frank et P. Wolfe, se résume comme suit:

Etape 0 : (Initialisation)
- Trouver une solution initiale x^1.
- Poser $\ell \longleftarrow 1$.

Etape ℓ : (Etape générale)

1. Sous-problème linéaire
   - Calculer y^ℓ tel que:
   - $y^\ell = \arg\Big\{\min\{\nabla\mathcal{F}(x^\ell)^T y \ : \ y \in D\}\Big\}$.
2. Direction de descente
   - Calculer $d^\ell = y^\ell - x^\ell$.
3. Test d'arrêt
   - Si un certain critère est vérifié,
     alors **ARRÊTER**.
4. Recherche linéaire
   - Trouver λ^ℓ tel que :
   - $\lambda^\ell = \arg\Big\{\min\{\mathcal{F}(x^\ell + \lambda \cdot d^\ell) : \lambda \in [0,1]\}\Big\}$.
5. Mise à jour des variables
   - Calculer : $x^{\ell+1} = x^\ell + \lambda^\ell \cdot d^\ell$.
   - Poser $\ell \longleftarrow \ell + 1$.
   - Retourner à l'étape ℓ.

Algorithme A: Algorithme de Frank et Wolfe

C'est dans la flexibilité du choix de l'approche de résolution du sous-problème linéaire à l'étape ℓ, que réside toute la souplesse de la méthode; et c'est là que se fait réellement tout le travail d'adaptation.

2. Le sous-problème linéaire

Notons momentanément par $\mathcal{F}(v,g)$ l'objectif (1) restreint au domaine des équations (2) à (6). Avec cette notation simplifiée nous pouvons écrire :

$$\mathcal{F}(v,g) = \sum_{a \in \mathcal{A}} F_a(v)$$

où $F_a(v) = v_a \cdot s_a(v_a)$. En effet, même si nous désirons connaître en même temps les flots optimaux du problème et la "meilleure" matrice de demande, il n'en demeure pas moins que le véritable objet du problème est la minimisation du coût total de transport, puisqu'il n'y a pas de coefficient pour la demande dans l'objectif. Cette remarque justifie que, par la suite, nous négligerions la composante de la demande dans le calcul du gradient de l'objectif.

Remarquons d'autre part que les contraintes du problème sont exprimées en termes du flot sur les chemins et non du flot sur les arcs; ceci n'est pas un problème en soi puisque l'équation (7) nous permet de passer d'une formulation à l'autre sans encombre. Cependant, puisque nous voulons trouver par la même occasion les entrées de la matrice O/D, c'est-à-dire le total du flot sur l'ensemble des chemins qui relient une origine o à une destination d — et ce pour toutes les combinaisons de paires (o,d) — il est de notre avantage, dans le sous-problème, de faire l'approximation linéaire par dérivation sur les flots des chemins plutôt que sur ceux des arcs. De sorte que l'on écrira:

$$\nabla_h \mathcal{F}(v,g) \;=\; \left(\frac{\partial}{\partial h_k}\mathcal{F}(v,g)\right)_{k \in K_{od}, o \in \mathcal{O}, d \in \mathcal{D}}$$

$$\begin{aligned}\nabla_h \mathcal{F}(v,g) &= \left(\frac{\partial}{\partial h_k}\left[\sum_{a\in\mathcal{A}} F_a(v)\right]\right)_{k\in K_{od},o\in\mathcal{O},d\in\mathcal{D}} \\ &= \left(\sum_{a\in\mathcal{A}} \frac{\partial}{\partial h_k} F_a(v)\right)_{k\in K_{od},o\in\mathcal{O},d\in\mathcal{D}}\end{aligned}$$

Plaçons-nous maintenant dans le contexte de l'exécution de l'algorithme. Supposons que $(\ell-1)$ itérations aient déjà été effectuées, et soit $\left(v^\ell, g^\ell\right)$ la solution courante.

En dérivant l'équation de définition (7) par rapport au flot du chemin h_k nous obtenons :

$$\frac{\partial v_a}{\partial h_k} = \delta_{ak}, \quad \forall a \in \mathcal{A}, \forall k \in K_{od}, \forall o \in \mathcal{O}, \forall d \in \mathcal{D}.$$

La règle de la dérivation en chaîne nous permet d'autre part d'écrire:

$$\frac{\partial F_a(v^\ell)}{\partial h_k} = \frac{\partial v_a}{\partial h_k} \cdot s_a(v_a^\ell) + v_a^\ell \cdot \frac{\partial v_a}{\partial h_k} \cdot s_a'(v_a^\ell),$$

ou encore :

$$\frac{\partial F_a(v^\ell)}{\partial h_k} = \delta_{ak} \cdot \left(s_a(v_a^\ell) + v_a^\ell \cdot s_a'(v_a^\ell)\right);$$

de sorte que l'objectif du sous-problème, à l'itération ℓ, s'exprimera:

$$\min \nabla_h \mathcal{F}\left(v^\ell, g^\ell\right)^T z,$$

ce qui est équivalent à:

$$\min \sum_{o\in\mathcal{O}} \sum_{d\in\mathcal{D}} \sum_{k\in K_{od}} \left\{\sum_{a\in\mathcal{A}} \delta_{ak} \cdot \left(s_a(v_a^\ell) + v_a^\ell \cdot s_a'(v_a^\ell)\right)\right\} \cdot z_k. \qquad (24)$$

Or, le sous-objectif (24), soumis aux contraintes (2) à (6), n'est autre que le problème de trouver les flots sur un réseau à coûts constants: les coûts marginaux de la solution courante. Pour simplifier nous noterons ces coûts, en tenant compte des deux représentations possibles du réseau:

$$\begin{aligned}S_a^\ell &= \left(s_a(v_a^\ell) + v_a^\ell \cdot s_a'(v_a^\ell)\right) \\ S_k^\ell &= \sum_{a\in\mathcal{A}} \delta_{ak} \cdot S_a^\ell\end{aligned}$$

Le sous-problème (24) s'écrira alors:

$$\min \sum_{o\in\mathcal{O}} \sum_{d\in\mathcal{D}} \sum_{k\in K_{od}} S_k^{\ell} \cdot z_k \tag{25}$$

sujet à :

$$-\sum_{k\in K_{od}} z_k + w_{od} = 0, \qquad \forall o \in \mathcal{O}, \forall d \in \mathcal{D} \tag{26}$$

$$\sum_{d\in\mathcal{D}} w_{od} - O_o = 0, \qquad \forall o \in \mathcal{O} \tag{27}$$

$$\sum_{o\in\mathcal{O}} w_{od} - D_d = 0, \qquad \forall d \in \mathcal{D} \tag{28}$$

$$z_k \geq 0, \qquad \forall k \in K_{od}, \forall o \in \mathcal{O}, \forall d \in \mathcal{D} \tag{29}$$

$$w_{od} \geq 0, \qquad \forall o \in \mathcal{O}, \forall d \in \mathcal{D} \tag{30}$$

$$y_a = \sum_{o\in\mathcal{O}} \sum_{d\in\mathcal{D}} \sum_{k\in K_{od}} \delta_{ak} \cdot z_k, \qquad \forall a \in \mathcal{A} \tag{31}$$

où z_k (ou y_a) et w_{od} sont les inconnues du sous-problème.

Puisque le flot v^{ℓ} est fixe à l'itération ℓ, les coûts S_k^{ℓ} à une itération donnée sont des constantes. La solution est donc immédiate, et consiste tout simplement à affecter le flot nécessaire à la satisfaction de la demande sur les chemins les plus courts en termes des coûts marginaux de la solution courante. En effet, soit S_{od}^{ℓ} la valeur du chemin le plus court reliant la paire de centroïdes (o, d); il est alors clair que:

$$\sum_{o\in\mathcal{O}} \sum_{d\in\mathcal{D}} \sum_{k\in K_{od}} S_k^{\ell} \cdot z_k \geq \sum_{o\in\mathcal{O}} \sum_{d\in\mathcal{D}} \sum_{k\in K_{od}} S_{od}^{\ell} \cdot z_k, \tag{32}$$

et par simple arithmétique, en se rappelant la contrainte (26) on peut écrire:

$$\sum_{o\in\mathcal{O}} \sum_{d\in\mathcal{D}} \sum_{k\in K_{od}} S_{od}^{\ell} \cdot z_k = \sum_{o\in\mathcal{O}} \sum_{d\in\mathcal{D}} S_{od}^{\ell} \cdot \sum_{k\in K_{od}} z_k = \sum_{o\in\mathcal{O}} \sum_{d\in\mathcal{D}} S_{od}^{\ell} \cdot w_{od}.$$

Le problème se réduit donc finalement au problème de transport classique (33) – (36), dont la solution ne pose aucune difficulté. Il existe du reste un

vaste choix d'algorithmes et de logiciels, comme RNET par exemple (voir [14]), permettant de le résoudre de façon efficace:

$$\min \sum_{o \in \mathcal{O}} \sum_{d \in \mathcal{D}} S^{\ell}_{od} \cdot w_{od} \tag{33}$$

sujet à :

$$\sum_{d \in \mathcal{D}} w_{od} = O_o, \qquad \forall o \in \mathcal{O} \tag{34}$$

$$\sum_{o \in \mathcal{O}} w_{od} = D_d, \qquad \forall d \in \mathcal{D} \tag{35}$$

$$w_{od} \geq 0, \qquad \forall o \in \mathcal{O}, \forall d \in \mathcal{D} \tag{36}$$

Rappelons-nous cependant que nous désirons trouver la solution (z^{ℓ}, w^{ℓ}) du problème linéarisé. Or, la résolution du problème (33) - (36) ne nous donne que la solution w^{ℓ}. Il reste donc à déterminer z^{ℓ} (et donc y^{ℓ}). Il suffit, pour cela de faire une affectation tout-ou-rien de la matrice w^{ℓ} aux chemins les plus courts (de coûts S^{ℓ}_{od}), puisque c'est ainsi que nous obtiendrons le moindre coût (se rappeler l'équation (32)).

3. La direction de diminution

Etant ainsi en possession de la solution optimale (z^{ℓ}, w^{ℓ}) de l'approximation linéaire de l'objectif du problème, nous sommes en mesure de définir la direction de diminution de la valeur de l'objectif où se trouve la nouvelle solution. Désignons par Δ^{ℓ} cette direction. Alors:

$$\Delta^{\ell} = (y^{\ell} - v^{\ell})$$

Notons encore une fois que le fait que la demande n'intervienne pas dans l'objectif nous rend libre de l'ignorer dans le calcul de Δ^{ℓ}, puisqu'il s'agit d'une direction selon laquelle se produit une diminution stricte de la valeur de l'objectif.

4. Le pas optimal

Il faut maintenant procéder à une recherche linéaire dans cette direction pour trouver le point qui, entre v^ℓ et y^ℓ, fait prendre à l'objectif sa valeur minimale. Nous allons pour cela résoudre le problème:

$$\min_{0\leq\lambda\leq 1} \sum_{a\in\mathcal{A}} \left\{(v_a^\ell + \lambda\cdot\Delta_a^\ell)\cdot s_a(v_a^\ell + \lambda\cdot\Delta_a^\ell)\right\}$$

où λ est le pas à effectuer dans la direction de diminution. L'objectif de ce problème est une fonction scalaire, en général non linéaire, convexe et continue sur l'intervalle $[0,1]$. Elle prend donc sa valeur minimale soit à l'une des bornes de l'intervalle, soit au point λ^ℓ où la dérivée s'annule. Analytiquement, le problème revient à résoudre l'équation à une seule inconnue suivante, en restreignant le domaine de solution à l'intervalle $[0,1]$:

$$\sum_{a\in\mathcal{A}} \left\{s_a(v_a^\ell + \lambda\cdot\Delta_a^\ell) + (v_a^\ell + \lambda\cdot\Delta_a^\ell)\cdot s_a'(v_a^\ell + \lambda\cdot\Delta_a^\ell)\right\}\cdot\Delta_a^\ell = 0$$

En pratique cependant, la recherche linéaire ne s'effectuera pas de façon analytique, mais plutôt à l'aide d'une des méthodes d'exploration numérique. Citons par exemple, la méthode de la bissection, de la section dorée, de Newton, les méthodes de lissage quadratique et cubique, ... *etc.* [17].

5. La mise à jour des variables

Le pas optimal λ^ℓ une fois trouvé, il ne reste plus qu'à mettre la solution à jour. Cette mise à jour est immédiate pour le flot:

$$v^{\ell+1} = v^\ell + \lambda^\ell\cdot\Delta^\ell \qquad \text{ou encore} \qquad v_a^{\ell+1} = v_a^\ell + \lambda^\ell\cdot\Delta_a^\ell \qquad (37)$$

Mais elle n'est guère plus compliquée pour la matrice O/D, comme nous allons le montrer:

Commençons en effet par exprimer la relation (37) en termes du flot sur les chemins, soit, pour un chemin quelconque k:

$$h_k^{\ell+1} = h_k^\ell + \lambda^\ell \cdot (z_k^\ell - h_k^\ell)$$

Considérons maintenant une certaine paire (o, d); le nouveau flot sur tous les chemins reliant o à d s'écrit:

$$\sum_{k \in K_{od}} h_k^{\ell+1} = \sum_{k \in K_{od}} h_k^\ell + \lambda^\ell \cdot \sum_{k \in K_{od}} (z_k^\ell - h_k^\ell),$$

ce qui, compte tenu des équations (2) et (26) de conservation du flot, revient à écrire:

$$g_{od}^{\ell+1} = g_{od}^\ell + \lambda^\ell \cdot (w_{od}^\ell - g_{od}^\ell) = (1 - \lambda^\ell) \cdot g_{od}^\ell + \lambda^\ell \cdot w_{od}^\ell,$$

relation très semblable à celle de la mise à jour des flots.

6. Le critère d'arrêt

Il n'existe pas de critère d'arrêt universel; en général, un critère donné présente à la fois des avantages et des inconvénients. Nous en présenterons trois.

Le premier consiste à spécifier un seuil d'amélioration arbitraire de la valeur de la dérivée directionnelle, lequel dépend du degré de précision souhaité, ε par exemple:

$$\text{Si } \left\| \nabla \mathcal{F}(v^\ell) \cdot \Delta^\ell \right\| \leq \varepsilon \text{ alors } \textbf{ARRÊTER}$$

Ce critère correspond à une double condition: soit que le vecteur définissant la direction de descente est très petit (entendre par là que la prochaine solution sera, dans le pire des cas, presque égale à la solution actuelle); soit que le gradient de l'objectif est pratiquement nul (entendre par là que nous sommes

au "voisinage" d'un minimum global). Dans tous les cas, cela indique un "plat" dans l'objectif. Mais il suffit alors d'imaginer un objectif décroissant très lentement à partir d'un point assez éloigné de l'optimum pour se rendre compte que ce critère puisse être insuffisant à lui seul.

Un second critère consiste à décider d'un nombre arbitraire d'itérations à effectuer. Cela suppose une très bonne connaissance *a priori* du problème, car l'on risque sinon, soit d'arrêter trop loin de l'objectif, et donc d'avoir une solution très sous-optimale, soit d'effectuer un nombre inutilement grand d'itérations parce que l'optimum aura été atteint plus vite que prévu. Il y a toujours, bien sûr, le compromis entre ces deux extrêmes.

L'écart relatif, appelé *gap* relatif, constitue le troisième critère. Il se défini comme suit:

$$gap \text{ relatif} = \frac{gap}{\mathcal{F}(v^\ell)}$$

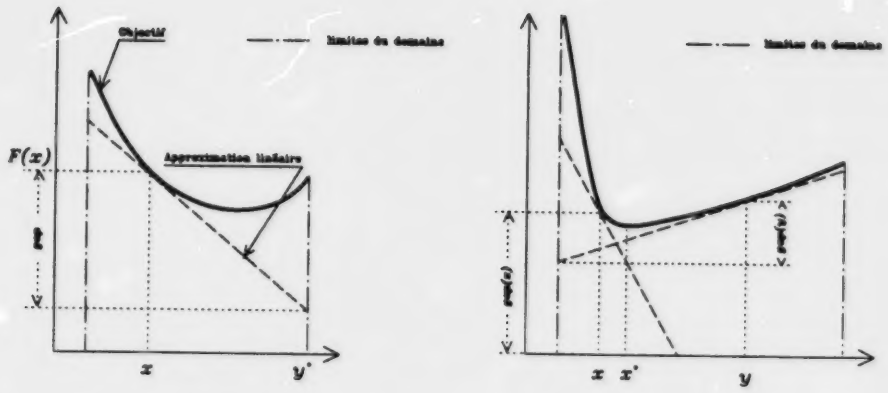
$$\text{où} \quad gap = \left(\mathcal{F}(v^\ell) - MBI^\ell\right)$$

Il permet de mesurer l'amélioration de la solution par rapport à MBI^ℓ, la meilleure borne inférieure disponible sur la valeur de l'optimum recherché. On peut en donner la définition récursive suivante:

$$\begin{aligned} MBI^0 &= -\infty \\ MBI^\ell &= \max\left\{\mathcal{F}(v^\ell) + \nabla\mathcal{F}(v^\ell)\cdot(y^\ell - v^\ell),\ MBI^{\ell-1}\right\}, \forall \ell \geq 1 \end{aligned}$$

Remarquons que $\mathcal{F}(v^\ell) + \nabla\mathcal{F}(v^\ell)\cdot(y^\ell - v^\ell)$, n'est autre que l'approximation linéaire de l'objectif au point $\left(v^\ell, \mathcal{F}(v^\ell)\right)$, dont la valeur minimale est atteinte en $y = y^\ell$ (voir figure 3-a).

Si ce dernier critère semble intéressant, il n'en demeure pas moins que les résultats obtenus restent seulement indicatifs, puisqu'ils dépendent beaucoup de la solution initiale choisie au début de l'exécution de l'algorithme et du



a- *gap* mesuré sur une fonction à une seule inconnue. Le *gap* relatif est le rapport de *gap* à $F(x)$.

b- Valeur indicative du *gap*: l'objectif en x est plus proche de l'optimum x^* qu'en y, mais le *gap* en y est plus petit.

Figure 3: Le *gap* d'une fonction à une seule inconnue.

cheminement suivi pour évoluer vers l'optimum, comme le met en relief la figure 3-b.

Les remarques précédentes incitent à s'appuyer sur plus d'un critère à la fois pour contrôler l'arrêt de la procédure. Nous avons pour notre part retenu les deux derniers en vue de les intégrer à l'algorithme de résolution.

7. L'initialisation

Ayant explicité tous les détails de l'étape générale de l'algorithme il ne reste plus qu'à déterminer comment générer la solution de départ.

L'usager dispose souvent d'une matrice basée sur des résultats empiriques, mais ce n'est pas toujours le cas. Comme, nous avons toujours besoin d'une

matrice de départ pour obtenir les flots de la solution initiale, il faut disposer d'une méthode qui permette de la créer numériquement. Diverses techniques peuvent être envisagées, puisque la procédure de Frank et Wolfe converge vers le minimum global indépendamment du point de départ choisi. Nous en présenterons deux: la première génère la matrice désirée à partir du flot nul, tandis que la deuxième le fait en utilisant une méthode de balancement.

7.1. Génération à partir d'un flot nul

Une façon simple de procéder consiste à résoudre le sous-problème linéaire seulement, en partant d'un flot initial nul ($v^0 = 0$); elle consiste à:

1- Calculer la valeur des plus courts chemins reliant chaque paire de centroïdes, en termes des coûts marginaux correspondant à un flot initial nul.

2- Résoudre le problème de transport où un seul arc relie chaque origine o à chaque destination d, avec un coût unitaire égal à la valeur du plus court chemin entre o et d calculé précédemment.

La solution w^0 ainsi trouvée donne une matrice initiale $g^1 = w^0$, et l'affectation de cette matrice aux chemins les plus courts donne à son tour les flots initiaux recherchés $v^1 = y^0$.

L'algorithme C ci-dessous résume cette première méthode:

1. Poser: $v^0 \longleftarrow 0$
2. Calculer $S^0_{od}, \forall o \in \mathcal{O}, \forall d \in \mathcal{D}$.
3. Résoudre le problème de transport.
   Soit w^0 la solution recherchée.
4. $g^1 \longleftarrow w^0$
5. Affecter g^1 au réseau pour obtenir v^1.

Algorithme C: Initialisation à partir d'un flot nul.

L'avantage de cette méthode est qu'elle s'intègre bien dans l'algorithme de Frank et Wolfe tel que nous l'avons adapté au problème (voir en fin de chapitre le résumé de toutes les étapes): elle ne nécessite en outre aucune procédure de calcul particulière qui ne serait pas utilisée dans l'étape générale.

7.2. Méthode de balancement à deux dimensions

Une autre technique de génération analytique de la matrice initiale, peut aussi être envisagée. Elle utilise uniquement l'information du réseau physique sous-jacent et les données constantes du problème. Cette technique est basée sur le modèle de distribution (ou modèle d'interaction spatiale):

$$\min \sum_{o\in\mathcal{O}} \sum_{d\in\mathcal{D}} g_{od} \left(\ln g_{od} + \theta \cdot u_{od}\right) \tag{38}$$

sujet à:

$$\sum_{d\in\mathcal{D}} g_{od} = \mathcal{O}_o, \qquad \forall o \in \mathcal{O} \tag{39}$$

$$\sum_{o\in\mathcal{O}} g_{od} = \mathcal{D}_d, \qquad \forall d \in \mathcal{D} \tag{40}$$

$$g_{od} \geq 0, \qquad \forall o \in \mathcal{O}, \forall d \in \mathcal{D} \tag{41}$$

où u_{od} est une mesure du coût et θ est un paramètre à calibrer. Nous savons que la solution de ce problème est de la forme:

$$\begin{aligned} g_{od} &= \exp\left(\alpha_o + \beta_d - 1\right) \cdot \exp\left(-\theta \cdot u_{od}\right) \\ &= r_o \cdot s_d \cdot \exp\left(-\theta \cdot u_{od}\right) \end{aligned}$$

où α_o et β_d sont des variables duales associées aux contraintes (39) et (40). Elle peut être facilement obtenue par la méthode de balancement dite de Furness [13]. Il s'agit cependant de choisir u_{od} et θ de façon adéquate.

Sur un réseau non congestionné où n'interviennent pas de capacités contraignantes et où le coût de transport varie en fonction de la distance parcourue, on aura naturellement tendance à distribuer l'offre vers les destinations, proportionnellement à l'inverse de la distance séparant les origines des destinations. Cette politique vise à satisfaire en priorité la demande des destinations les plus proches. Ainsi, en fixant $u_{od} = 1/dist(o,d)$ (où $dist(o,d)$ est la plus courte distance séparant o de d), et $\theta = 1$ l'algorithme s'écrit:

Etape 0:

Calculer: $g_{od}^0 = \exp(-1/dist(o,d)), \forall o \in \mathcal{O}, \forall d \in \mathcal{D}$.

Poser: $r_o^0 = 1, \forall o \in \mathcal{O}, s_d^0 = 1, \forall d \in \mathcal{D}$ et $i = 1$.

Etape i:

a- Calculer $r_o^{i+1} = \dfrac{\mathcal{O}_o}{\sum_{d \in \mathcal{D}} g_{od}^i} \cdot r_o^i, \forall o \in \mathcal{O}$.

b- Calculer $g_{od}^{i+\frac{1}{2}} = \dfrac{\mathcal{O}_o}{\sum_{d \in \mathcal{D}} g_{od}^i} \cdot g_{od}^i, \forall o \in \mathcal{O}, \forall d \in \mathcal{D}$.

c- Calculer $s_d^{i+1} = \dfrac{\mathcal{D}_d}{\sum_{o \in \mathcal{O}} g_{od}^{i+\frac{1}{2}}} \cdot s_d^i, \forall d \in \mathcal{D}$.

d- Calculer $g_{od}^{i+1} = \dfrac{\mathcal{D}_d}{\sum_{o \in \mathcal{O}} g_{od}^{i+\frac{1}{2}}} \cdot g_{od}^{i+\frac{1}{2}}, \forall o \in \mathcal{O}, \forall d \in \mathcal{D}$.

e- Si $\|r_o^{i+1} - r_o^i\| \leq \varepsilon$ et $\|s_d^{i+1} - s_d^i\| \leq \varepsilon$ alors **ARRÊTER**.

f- Sinon poser: $i \longleftarrow i + 1$ et retourner en **a-**.

Algorithme D: Initialisation par balancement.

L'avantage de cette deuxième manière de faire est qu'en modifiant la métrique de coût u_{od} et le paramètre θ, on peut générer plusieurs points de départ pour l'algorithme, ce qui permet d'examiner la convergence de l'algorithme vers des solutions ε-optimales. C'est donc une approche très utile pour trouver une bonne solution de départ pour un réseau dont les coûts seraient fortement non linéaires.

En guise de conclusion, l'algorithme B ci-dessous résume toutes les étapes du développement précédent.

Etape 0: (Initialisation)

Trouver une solution initiale (v^1, g^1) en adaptant convenablement l'algorithme C ou l'algorithme D.

Poser $\ell \longleftarrow 1$.

Etape ℓ: (Etape générale)

1. Sous-problème linéaire
   - Calculer les coûts S^ℓ_{od} des chemins k^* les plus courts en termes du flot courant v^ℓ.
   - Résoudre le problème de transport (33) – (36). La solution optimale est w^ℓ.
   - Faire l'affectation tout-ou-rien de la matrice w au réseau pour obtenir z^ℓ_k, puis, par (31), y^ℓ_a.
2. Direction de descente
   - Calculer $\Delta^\ell = y^\ell - v^\ell$
3. Test d'arrêt
   - Calculer le *gap* de l'itération ℓ.

   SI $\left\{ \begin{array}{c} \text{nombre maximal d'itérations atteint} \\ \text{ou } \ gap \leq \varepsilon_{gap} \end{array} \right\}$ ARRÊTER
4. Recherche Linéaire
   - A l'aide d'une des méthodes numériques d'exploration, trouver λ^ℓ tel que:

     $\lambda^\ell = \arg\left\{\min_{0\leq\lambda\leq 1} \sum_{a\in\mathcal{A}} (v^\ell_a + \lambda\cdot\Delta^\ell_a)\cdot s_a(v^\ell_a + \lambda\cdot\Delta^\ell_a)\right\}$
5. Mise à jour des variables
   - $v^{\ell+1} = v^\ell + \lambda^\ell\cdot\Delta^\ell$
   - $g^{\ell+1} = (1-\lambda^\ell)\cdot g^\ell + \lambda^\ell\cdot w^\ell$
   - Poser $\ell \longleftarrow \ell + 1$
   - Retourner à l'étape ℓ.

Algorithme B: Algorithme pour le problème de base.

Chapitre III Généralisation de l'approche

Tout au long de ce chapitre, et pour alléger le texte, le signe (†) remplacera la série d'appartenance suivante: $o \in \mathcal{O}, d \in \mathcal{D}, m(p) \in \mathcal{M}(p), p \in \mathcal{P}$.

Au chapitre précédent, nous avons développé l'algorithme tel qu'il faudrait l'appliquer pour résoudre la version la plus simple du problème. Pour compléter notre étude théorique nous allons maintenant montrer comment modifier l'algorithme B pour tenir compte d'une complexité accrue. Cette dernière se manifeste dans la multiplication des produits transportés combinée à l'introduction des regroupements de modes. En effet, la multimodalité seule, ne suffit pas à modifier le degré de complexité du problème, puisque, comme déjà souligné au chapitre I, rien ne différencie le modèle unimodal du modèle multimodal sinon l'interprétation des ensembles $\mathcal{A}$ et K_{od}.

Cette généralisation est cependant loin d'être trivial. En effet, le réseau de transport n'étant plus unique, parler de coûts marginaux et de chemins les plus courts en termes de ces coûts n'a de sens que s'ils sont considérés en rapport avec un produit p et le sous-réseau sur lequel il est transporté — cela, d'autant plus que chaque produit à ses fonctions de coût propres.

Examinons le problème linéaire pour commencer. Soit $k \in K_{od}^{m(p)}$ un

chemin dans le sous-réseau induit par un regroupement de modes $m(p)$ du produit p; plaçons-nous à l'itération ℓ du processus de résolution. Le gradient de l'objectif s'écrit alors:

$$\begin{aligned}
\nabla_h \mathcal{F}(v^\ell) &= \left(\frac{\partial}{\partial h_k}\mathcal{F}(v^\ell)\right)_{k\in K_{od}^{m(p)},\dagger} \\
&= \left(\frac{\partial}{\partial h_k}\left[\sum_{\pi\in\mathcal{P}}\sum_{a\in\mathcal{A}} v_a^{\pi\ell}\cdot s_a^\pi(v_a^\ell)\right]\right)_{k\in K_{od}^{m(p)},\dagger} \\
&= \left(\sum_{\pi\in\mathcal{P}}\sum_{a\in\mathcal{A}}\frac{\partial}{\partial h_k}\left[v_a^{\pi\ell}\cdot s_a^\pi(v_a^\ell)\right]\right)_{k\in K_{od}^{m(p)},\dagger}
\end{aligned}$$

Or, en tenant compte de la règle de dérivation en chaîne, nous pouvons écrire:

$$\begin{aligned}
\frac{\partial}{\partial h_k}\left(v_a^{\pi\ell}\cdot s_a^\pi(v_a^\ell)\right) &= \frac{\partial}{\partial v_a^p}\left(v_a^{\pi\ell}\cdot s_a^\pi(v_a^\ell)\right)\cdot\frac{\partial v_a^p}{\partial h_k} \\
&= \begin{cases} \left(v_a^{\pi\ell}\cdot s_a'^\pi(v_a^\ell)+s_a^\pi(v_a^\ell)\right)\cdot\delta_{ak}, & \text{si} \quad \pi = p \\ \left(v_a^{\pi\ell}\cdot s_a'^\pi(v_a^\ell)\right)\cdot\delta_{ak}, & \text{si} \quad \pi \neq p \end{cases}
\end{aligned}$$

D'où:

$$\nabla_h \mathcal{F}(v^\ell) = \left(\sum_{a\in\mathcal{A}}\delta_{ak}\cdot\left\{\sum_{\pi\in\mathcal{P}}\left(v_a^{\pi\ell}\cdot s_a'^\pi(v_a^\ell)\right)+s_a^p(v_a^\ell)\right\}\right)_{k\in K_{od}^{m(p)},\dagger}$$

C'est le vecteur des coûts du sous-problème linéaire. Nous remarquons d'emblée que ces derniers sont différents de ceux trouvés pour le problème de base. En effet, au coût marginal du transport du produit p sur le chemin k s'ajoutent désormais des termes croisés. Ils représentent l'impact du transport concurrentiel, sur ce même chemin, des autres produits du problème.

L'expression de ce coût étant un peu lourde, nous adopterons la notation suivante, tout en gardant à l'esprit la distinction de signification qui s'impose dans le contexte:

$$\begin{aligned}
S_a^{m(p)\ell} &= \sum_{\pi\in\mathcal{P}}\left(v_a^{\pi\ell}\cdot s_a'^\pi(v_a^\ell)\right)+s_a^p(v_a^\ell) \\
S_k^{m(p)\ell} &= \sum_{a\in\mathcal{A}}\delta_{ak}\cdot S_a^{m(p)\ell}
\end{aligned}$$

Le gradient de l'objectif s'écrit alors:

$$\nabla_h \mathcal{F}(v^\ell) = \left(\sum_{a \in \mathcal{A}} \delta_{ak} \cdot S_a^{m(p)\ell} \right) = \left(S_k^{m(p)\ell} \right)_{k \in K_{od}^{m(p)}, \dagger}$$

et le sous-problème linéaire devient:

$$\min \nabla_h \mathcal{F}(v^\ell)^T z$$

$$\Vert\Vert$$

$$\min \sum_{p \in \mathcal{P}} \sum_{m(p) \in \mathcal{M}(p)} \sum_{o \in \mathcal{O}} \sum_{d \in \mathcal{D}} \sum_{k \in K_{od}^{m(p)}} S_k^{m(p)\ell} \cdot z_k \tag{42}$$

sujet à:

$$- \sum_{k \in K_{od}^{m(p)}} z_k + w_{od}^{m(p)} = 0, \qquad \forall o \in \mathcal{O}, \forall d \in \mathcal{D}, \forall m(p) \in \mathcal{M}(p)^{[1]} \tag{43}$$

$$\sum_{m(p) \in \mathcal{M}(p)} \sum_{d \in \mathcal{D}} w_{od}^{m(p)} = O_o^p, \qquad \forall o \in \mathcal{O}^{[1]} \tag{44}$$

$$\sum_{m(p) \in \mathcal{M}(p)} \sum_{o \in \mathcal{O}} w_{od}^{m(p)} = D_d^p, \qquad \forall d \in \mathcal{D}^{[1]} \tag{45}$$

$$z_k \geq 0, \qquad \forall k \in K_{od}^{m(p)}, \qquad \forall o \in \mathcal{O}, \forall d \in \mathcal{D}, \forall m(p) \in \mathcal{M}(p)^{[1]} \tag{46}$$

$$w_{od}^{m(p)} \geq 0, \qquad \forall o \in \mathcal{O}, \forall d \in \mathcal{D}, \forall m(p) \in \mathcal{M}(p)^{[1]} \tag{47}$$

$$y_a = \sum_{p \in \mathcal{P}} y_a^p, \qquad \forall a \in \mathcal{A} \tag{48}$$

$$y_a^p = \sum_{m(p) \in \mathcal{M}(p)} \sum_{o \in \mathcal{O}} \sum_{d \in \mathcal{D}} \sum_{k \in K_{od}^{m(p)}} \delta_{ak} \cdot z_k, \quad \forall a \in \mathcal{A}^{[1]} \tag{49}$$

[1] $\forall p \in \mathcal{P}$

Considérons maintenant un certain produit p et un regroupement de modes $m(p)$. Pour chacun des choix p et $m(p)$, et pour chaque paire de centroïdes (o, d), nous pouvons trouver le chemin de coût minimal entre o et d et noter par $S_{od}^{m(p)\ell}$ la valeur de ce coût:

$$S_{od}^{m(p)\ell} = \min \left\{ S_k^{m(p)\ell} : k \in K_{od}^{m(p)} \right\}, \forall o \in \mathcal{O}, \forall d \in \mathcal{D}, \forall m(p) \in \mathcal{M}(p), \forall p \in \mathcal{P}.$$

Il est facile de voir que:

$$\sum_{o\in\mathcal{O}}\sum_{d\in\mathcal{D}}\sum_{k\in K_{od}^{m(p)}} S_k^{m(p)\ell}\cdot z_k \geq \sum_{o\in\mathcal{O}}\sum_{d\in\mathcal{D}} S_{od}^{m(p)\ell}\cdot w_{od}^{m(p)},$$

et le problème se ramène en conséquence à l'expression considérablement simplifiée:

$$\min \sum_{p\in\mathcal{P}}\sum_{o\in\mathcal{O}}\sum_{d\in\mathcal{D}}\left\{\sum_{m(p)\in\mathcal{M}(p)} S_{od}^{m(p)\ell}\cdot w_{od}^{m(p)}\right\} \tag{50}$$

sujet à:

$$\sum_{m(p)\in\mathcal{M}(p)}\sum_{d\in\mathcal{D}} w_{od}^{m(p)} = O_o^p, \quad \forall o\in\mathcal{O},\forall p\in\mathcal{P} \tag{51}$$

$$\sum_{m(p)\in\mathcal{M}(p)}\sum_{o\in\mathcal{O}} w_{od}^{m(p)} = D_d^p, \quad \forall d\in\mathcal{D},\forall p\in\mathcal{P} \tag{52}$$

$$w_{od}^{m(p)} \geq 0, \quad \forall o\in\mathcal{O},\forall d\in\mathcal{D},\forall m(p)\in\mathcal{M}(p),\forall p\in\mathcal{P} \tag{53}$$

Avec cette dernière formulation, nous avons un problème séparable en une série de $|\mathcal{P}|$ problèmes individels similaires, relatifs chacun à l'un des produits transportés:

$$\min \sum_{o\in\mathcal{O}}\sum_{d\in\mathcal{D}}\left\{\sum_{m(p)\in\mathcal{M}(p)} S_{od}^{m(p)\ell}\cdot w_{od}^{m(p)}\right\} \tag{54}$$

sujet à:

$$\sum_{m(p)\in\mathcal{M}(p)}\sum_{d\in\mathcal{D}} w_{od}^{m(p)} = O_o^p, \quad \forall o\in\mathcal{O} \tag{55}$$

$$\sum_{m(p)\in\mathcal{M}(p)}\sum_{o\in\mathcal{O}} w_{od}^{m(p)} = D_d^p, \quad \forall d\in\mathcal{D} \tag{56}$$

$$w_{od}^{m(p)} \geq 0, \quad \forall o\in\mathcal{O},\forall d\in\mathcal{D},\forall m(p)\in\mathcal{M}(p) \tag{57}$$

Contrairement au problème (33) – (36), il ne s'agit pas d'un problème de transport classique. Pour l'analyser, considérons un petit réseau simple avec 3 origines, 2 destinations et 2 sous-réseaux de distribution par produit distribué: m_1^p et m_2^p.

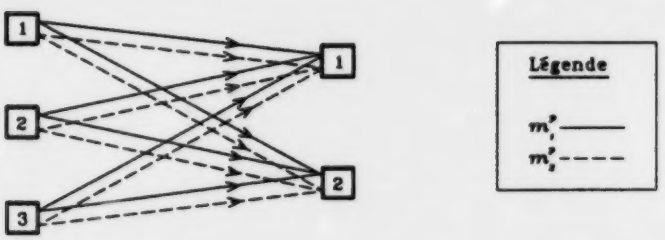


Figure 4: Réseau figurant le problème (54) – (57).

Dans la figure 4 ci-dessus, nous voyons que la dernière formulation correspondrait à un réseau où chaque paire (o,d) serait reliée par une série de liens parallèles, un pour chacun des chemins de coût minimal $S_{od}^{m(p)\ell}$, ayant justement $S_{od}^{m(p)\ell}$ comme coût unitaire de transport. Notons que si une paire (o,d) n'est pas reliée dans l'un des sous-réseaux $m(p)$, il suffit de lui associer un lien de coût infini pour ce sous-réseau.

Il est clair que si nous étions capables de supprimer tous les arcs parallèles sauf un pour chaque paire de centroïdes, nous retrouverions un problème de transport classique. Remarquons maintenant que, pour toute paire (o,d), le coût minimal pour le transport de w_{od}^{p} est atteint en transportant l'intégralité de cette quantité sur le lien de coût minimal que nous noterons $S_{od}^{p\ell}$ (où $w_{od}^{p} = \sum_{m(p)\in\mathcal{M}(p)} w_{od}^{m(p)}$). De sorte que la solution à moindre coût du problème (54) – (57) est identique à celle du problème de transport classique où chaque paire (o,d) est reliée par un lien unique de coût:

$$S_{od}^{p\ell} = \min\left\{S_{od}^{m(p)\ell} : m(p) \in \mathcal{M}(P)\right\}.$$

Cette argumentation intuitive peut être renforcée par un raisonnement plus rigoureux. Pour cela, examinons la matrice des contraintes du programme linéaire (54) – (57) appliqué à l'exemple précédent:

	$w_{11}^{m_1^p}$	$w_{12}^{m_1^p}$	$w_{21}^{m_1^p}$	$w_{22}^{m_1^p}$	$w_{31}^{m_1^p}$	$w_{32}^{m_1^p}$	$w_{11}^{m_2^p}$	$w_{12}^{m_2^p}$	$w_{21}^{m_2^p}$	$w_{22}^{m_2^p}$	$w_{31}^{m_2^p}$	$w_{32}^{m_2^p}$
O_1	1	1	0	0	0	0	1	1	0	0	0	0
O_2	0	0	1	1	0	0	0	0	1	1	0	0
O_3	0	0	0	0	1	1	0	0	0	0	1	1
D_1	1	0	1	0	1	0	1	0	1	0	1	0
D_2	0	1	0	1	0	1	0	1	0	1	0	1

Dans cette matrice, une colonne correspondant à $w_{od}^{m_i^p}$ est linéairement dépendante de chaque colonne "homologue" (c'est-à-dire, relative à la même paire (o,d)) dans les autres sous-réseaux m_j^p. Par conséquent une, et une seule, des colonnes $w_{od}^{m(p)}$, $m(p) \in \mathcal{M}(p)$ pourra se trouver dans une solution de base. De plus, la colonne sélectionnée pour la recherche de la solution optimale étant celle de moindre coût, du fait de la dépendance linéaire déjà mentionnée, ce sera toujours celle de coût $S_{od}^{p\ell}$ qui sera sélectionnée.

Par conséquent, la solution du problème (54) – (57) revient à celle du problème suivant:

$$\min \sum_{o \in \mathcal{O}} \sum_{d \in \mathcal{D}} S_{od}^{p\ell} \cdot w_{od}^{p} \tag{58}$$

sujet à:

$$\sum_{d \in \mathcal{D}} w_{od}^{p} = O_o^p, \qquad \forall o \in \mathcal{O} \tag{59}$$

$$\sum_{o \in \mathcal{O}} w_{od}^{p} = D_d^p, \qquad \forall d \in \mathcal{D} \tag{60}$$

$$w_{od}^{p} \geq 0, \qquad \forall o \in \mathcal{O}, \forall d \in \mathcal{D} \tag{61}$$

Nous obtenons ainsi la solution $w^{p\ell}$ à partir de laquelle il est facile de reconstituer toutes les matrices $w^{m(p)\ell}$. Il suffit en effet, d'avoir conservé au préalable l'indice du sous-réseau $m(p)$ où se produit le coût minimal pour chaque paire (o,d) dans une matrice de taille $|\mathcal{O}| \times |\mathcal{D}|$, et cela, pour chacun des $|\mathcal{P}|$ produits à transporter.

Ayant les matrices $w^{m(p)\ell}$, il suffit de les affecter aux chemins les plus courts déjà calculés du sous-réseau $m(p)$ pour trouver la valeur des flots z^ℓ correspondant à la solution optimale de l'approximation linéaire. Ce dernier résultat permet de déterminer, pour chaque produit p, et chaque arc a, une direction de dimininution stricte du coût de transport de ce produit:

$$\Delta_a^{p\ell} = y_a^{p\ell} - v_a^{p\ell},$$

de sorte que la direction de diminution globale pour chaque arc s'écrit comme suit:

$$\Delta_a^\ell = \sum_{p \in \mathcal{P}} \Delta_a^{p\ell}.$$

Le pas optimal sera alors l'argument λ^ℓ qui minimise l'objectif le long de la direction $\Delta^\ell = \left(\Delta^{1\ell}, \ldots, \Delta^{|\mathcal{P}|\ell}\right)$, c'est-à-dire:

$$\lambda^\ell = \arg \left\{ \min_{0 \le \lambda \le 1} \sum_{p \in \mathcal{P}} \sum_{a \in \mathcal{A}} (v_a^{p\ell} + \lambda \cdot \Delta_a^{p\ell}) \cdot s_a^p (v_a^\ell + \lambda \cdot \Delta_a^\ell) \right\}$$

Enfin, la mise à jour des variables se fera, pour chaque produit individuellement, de la même façon que dans l'algorithme B.

L'algorithme E à la page suivante résume l'ensemble de la démarche.

Etape 0: (Initialisation)

Trouver une solution initiale (v^1, g^1) en adaptant comme il faut l'algorithme C ou l'algorithme D.

Poser $\ell \longleftarrow 1$.

Etape ℓ: (Etape générale)

1. Sous-problème linéaire

   Pour chaque produit p:

   * Calculer les coûts $S_{od}^{m(p)\ell}$ des chemins k^* les plus courts en termes du flot courant v^ℓ.
   * Trouver et retenir les coûts $S_{od}^{p\ell}$.
   * Résoudre le problème de transport (58) – (61). La solution optimale est $w^{p\ell}$.
   * Reconstituer les matrices $w^{m(p)\ell}$.
   * Effectuer l'affectation tout-ou-rien des nouvelles matrices $w_{od}^{m(p)\ell}, \forall p \in \mathcal{P}$ au réseau pour obtenir $z^{p\ell}$, puis $y^{p\ell}$, grâce à (49).

2. Direction de descente
   * Calculer $\Delta^{p\ell} = y^{p\ell} - v^{p\ell}, \forall p \in \mathcal{P}$.

3. Test d'arrêt
   * Calculer le *gap* de l'itération ℓ.

   SI $\left\{ \begin{array}{c} \text{nombre maximal d'itérations atteint} \\ \text{ou} \quad gap \leq \varepsilon_{gap} \end{array} \right\}$ ARRÊTER.

4. Recherche Linéaire

   A l'aide d'une des méthodes numériques d'exploration, trouver λ^ℓ tel que:

   $\lambda^\ell = \arg \left\{ \min_{0 \leq \lambda \leq 1} \sum_{p \in \mathcal{P}} \sum_{a \in \mathcal{A}} (v_a^{p\ell} + \lambda \cdot \Delta_a^{p\ell}) \cdot s_a^p (v_a^\ell + \lambda \cdot \Delta_a^\ell) \right\}$

5. Mise à jour des variables
   * $v^{p\ell+1} = v^{p\ell} + \lambda^\ell \cdot \Delta^{p\ell}, \forall p \in \mathcal{P}$.
   * $g^{m(p)\ell+1} = (1 - \lambda^\ell) \cdot g^{m(p)\ell} + \lambda^\ell \cdot w^{m(p)\ell}$.
   * Poser $\ell \longleftarrow \ell + 1$.
   * Retourner à l'étape ℓ.

Algorithme E: Algorithme pour le problème multiproduit.

Chapitre IV Cas particuliers

Au cours du développement théorique qui a précédé, nous avons mentionné l'existence de cas particuliers. Nous allons maintenant développer trois situations dont l'intérêt n'est pas des moindres, puisqu'elles mènent à une simplification considérable du processus de résolution. Dans un premier temps, nous nous arrêterons à l'impact d'une structure de coût linéaire de l'objectif; puis, nous examinerons deux alternatives à la formulation C du problème: une simplification par suppression d'un degré de complexité, et une autre par addition de données supplémentaires. L'une, comme l'autre mènent à une version de l'algorithme de résolution beaucoup plus proche de l'algorithme B que de l'algorithme E.

1. Réseau à coûts unitaires constants

Alors que la méthode d'approximation linéaire converge de façon sous-linéaire au voisinage de l'optimum, il est évident qu'en présence d'une structure linéaire de l'objectif du problème, la procédure converge **à l'optimum** en **au plus** une seule itération complète de l'étape générale. Cet état de fait est confirmé par le calcul du *gap* relatif.

En effet, pour chaque arc du réseau la fonction de coût s'écrit $s_a^p(v_a) = \gamma_a^p$

où γ_a^p est une constante réelle, de sorte que $s_a'^p(v_a) = 0, \forall a \in \mathcal{A}, \forall p \in \mathcal{P}$. Soit v_a^ℓ un point quelconque du domaine des flots réalisables où on évalue l'approximation linéaire de l'objectif.

$$\begin{aligned}
\nabla_h \mathcal{F}(v^\ell) &= \left(\sum_{a \in \mathcal{A}} \delta_{ak} \cdot \left\{ \sum_{\pi \in \mathcal{P}} \left(v_a^{\pi \ell} \cdot s_a'^{\pi}(v_a^\ell) \right) + s_a^p(v_a^\ell) \right\} \right) \\
&= \left(\sum_{a \in \mathcal{A}} \delta_{ak} \cdot s_a^p(v_a^\ell) \right) \\
&= \left(\sum_{a \in \mathcal{A}} \delta_{ak} \cdot \gamma_a^p \right)
\end{aligned}$$

Où $k \in K_{od}^{m(p)}, o \in \mathcal{O}, d \in \mathcal{D}, m(p) \in \mathcal{M}(p), p \in \mathcal{P}$. Le gradient de l'objectif est donc un vecteur constant, indépendant du point du domaine où il est évalué.

En nous rappelant la définition $\mathcal{K} = \bigcup_{m(p) \in \mathcal{M}(p)} \bigcup_{o \in \mathcal{O}} \bigcup_{d \in \mathcal{D}} K_{od}^{m(p)}$, examinons maintenant l'approximation linéaire; nous obtenons:

$$\begin{aligned}
\nabla_h \mathcal{F}(v^\ell)^T z &= \sum_{p \in \mathcal{P}} \sum_{a \in \mathcal{A}} \sum_{k \in \mathcal{K}} \delta_{ak} \cdot \gamma_a^p \cdot z_k \\
&= \sum_{p \in \mathcal{P}} \sum_{a \in \mathcal{A}} \gamma_a^p \cdot y_a^p \\
&= \mathcal{F}(y)
\end{aligned}$$

Ainsi, l'approximation linéaire de l'objectif n'est autre que l'objectif lui-même. Si bien que, dès la première itération, la solution optimale (z^*, w^*) du sous-problème linéaire évaluée en v^1 est aussi celle du problème maître, puisqu'ils sont identiques. Il en découle naturellement que le pas optimal pour la mise à jour des variables du problème a une valeur de 1.

Nous avons dit plus haut que la procédure convergeait en, au plus, une seule itération complète de l'étape générale. En effet, tout dépend de la solution initiale:

Si cette dernière est la solution optimale, alors l'itération 1 ne sera pas

complétée car le *gap* sera nul. En effet, la borne inférieure est la suivante:

$$\begin{aligned} MBI^1 &= \max\left(\mathcal{F}(v^1) + \nabla_{\Lambda}\mathcal{F}(v^1)\cdot(y^{1*} - v^1), -\infty\right) \\ &= \max\left(\mathcal{F}(v^1) + \mathcal{F}(y^{1*}) - \mathcal{F}(v^1), -\infty\right) \\ &= \mathcal{F}(y^{1*}) \end{aligned}$$

Mais comme $\mathcal{F}(v^1) = \mathcal{F}(y^{1*})$, on obtient:

$$gap = \frac{\mathcal{F}(v^1) - MBI^1}{\mathcal{F}(v^1)} = 0.00\%$$

et le *gap* confirme bien l'optimalité.

Si la solution de départ est quelconque, alors après la mise à jour nous aurons $\mathcal{F}(v^2) = \mathcal{F}(y^{1*})$. Le calcul de la borne inférieure donnera:

$$MBI^2 = \max\left(\mathcal{F}(y^{2*}), \mathcal{F}(v^2)\right)$$

Et comme $\mathcal{F}(y^{2*}) = \mathcal{F}(v^2)$ nous retrouvons un *gap* nul, et ceci confirme l'optimalité, de sorte que l'itération 2 ne sera pas complétée.

2. Un seul regroupement de modes par produit

Dans la formulation C, ce qui compliquait le modèle au maximum, c'était l'introduction de la notion de regroupement de modes en relation avec le transport d'un produit, cela pour les besoins d'une analyse plus détaillée du réseau. Ce n'est certes pas le regroupement en soi qui augmentait le degré de complexité du problème, mais le fait qu'à chacun d'eux l'on associait une matrice de demande différente $g^{m(p)}$, sans disposer pour autant d'aucune information sur ces matrices autre que leur seule existence.

Supposons donc que nous supprimions cette difficulté. C'est-à-dire, supposons que pour chaque produit, il n'y ait qu'un seul sous-réseau de distribution. La formulation C elle-même se simplifie, puisque nous pouvons alors

laisser tomber tous les indices $m(p)$ et les remplacer par l'indice p, de sorte que les équations (18) et (19) prennent l'allure plus familière suivante:

$$\sum_{d\in\mathcal{D}} g^p_{od} = O^p_o, \quad \forall o \in \mathcal{O}, \forall p \in \mathcal{P} \tag{62}$$

$$\sum_{o\in\mathcal{O}} g^p_{od} = D^p_d, \quad \forall d \in \mathcal{D}, \forall p \in \mathcal{P} \tag{63}$$

Ce changement n'influence que le développement et la résolution du sous-problème linéaire. Cette influence (mise à part la différence de notation) ne devient évidente dans le développement mathématique qu'au moment de l'entrée en jeu de la séparabilité par produit du problème, c'est-à-dire au niveau de la formulation (54) – (57) du problème à résoudre. Réécrivons ce dernier avec les amendements dûs à la nouvelle notation:

$$\min \sum_{o\in\mathcal{O}} \sum_{d\in\mathcal{D}} S^{p\ell}_{od} \cdot w^p_{od} \tag{64}$$

sujet à:

$$\sum_{d\in\mathcal{D}} w^p_{od} = O^p_o, \quad \forall o \in \mathcal{O}, \forall p \in \mathcal{P} \tag{65}$$

$$\sum_{o\in\mathcal{O}} w^p_{od} = D^p_d, \quad \forall d \in \mathcal{D}, \forall p \in \mathcal{P} \tag{66}$$

$$w^p_{od} \geq 0, \quad \forall o \in \mathcal{O}, \forall d \in \mathcal{D} \tag{67}$$

Nous retrouvons un problème de transport classique que l'on peut résoudre par des algorithmes connus pour ce problème. Nous obtenons ainsi très facilement la solution optimale $w^{p\ell *}$, sans la coûteuse nécessité d'utiliser des matrices additionnelles en guise de mémoire ponctuelle comme c'était le cas quand il y avait regroupement de modes selon la formulation C. Il suffit alors, comme dans l'algorithme B, de faire l'affectation tout-ou-rien de $w^{p\ell *}$ aux chemins les plus courts pour chaque paire (o, d) pour trouver les flots optimaux du problème approximé.

Ce cas est donc tout à fait similaire à celui du problème de base. A la différence près des coûts du problème de transport sous-jacent (mais il s'agit là d'un problème numérique et non méthodologique), la procédure de résolution est en fin de compte la même.

3. Information supplémentaire sur les $w^{m(p)}$

Supposons maintenant, que nous ayons à résoudre le problème avec le degré de complexité de la formulation C, mais que nous disposions, en plus, des totaux partiels de l'offre et de la demande relatifs à chaque regroupement de modes $m(p)$, pour chaque produit p. Notons-les respectivement $O_o^{m(p)}$, et $D_d^{m(p)}$. Nous allons montrer que le problème n'est pas plus compliqué avec, que sans regroupements de modes: tout se passe, au niveau des contraintes, comme si, au lieu d'un seul produit p, nous en avions $|\mathcal{M}(p)|$, et que pour chacun d'eux, nous disposions de l'offre globale, de la demande globale, et du réseau de distribution avec sa structure de coûts.

Commençons par mettre en évidence le changement que cela implique pour la formulation C. Rajoutons d'abord les équations qui traduisent cette information supplémentaire.

$$\sum_{d \in \mathcal{D}} w_{od}^{m(p)} = O_o^{m(p)}, \quad \forall o \in \mathcal{O}, \forall m(p) \in \mathcal{M}(p), \forall p \in \mathcal{P} \tag{68}$$

$$\sum_{o \in \mathcal{O}} w_{od}^{m(p)} = D_d^{m(p)}, \quad \forall d \in \mathcal{D}, \forall m(p) \in \mathcal{M}(p), \forall p \in \mathcal{P} \tag{69}$$

Les équations (18) et (19) de la formulation C deviennent alors redondantes et nous pouvons les supprimer.

Comme précédemment, seule la résolution du sous-problème linéaire est influencée par ces modifications. Cette influence se manifeste au niveau de

l'entrée en jeu de la séparabilité en $|\mathcal{P}|$ problèmes similaires. Si nous réécrivons le problème (54) – (57) avec ces nouvelles données, nous obtenons:

$$\min \sum_{o\in\mathcal{O}} \sum_{d\in\mathcal{D}} \sum_{m(p)\in\mathcal{M}(p)} S_{od}^{m(p)\ell} \cdot w_{od}^{m(p)} \tag{70}$$

sujet à:

$$\sum_{d\in\mathcal{D}} w_{od}^{m(p)} = O_o^{m(p)}, \qquad \forall o \in \mathcal{O}, \forall m(p) \in \mathcal{M}(p) \tag{71}$$

$$\sum_{o\in\mathcal{O}} w_{od}^{m(p)} = D_d^{m(p)}, \qquad \forall d \in \mathcal{D}, \forall m(p) \in \mathcal{M}(p) \tag{72}$$

$$w_{od}^{m(p)} \geq 0, \qquad \forall o \in \mathcal{O}, \forall d \in \mathcal{D}, \forall m(p) \in \mathcal{M}(p) \tag{73}$$

Ce dernier problème est, lui aussi, séparable, mais cette fois en $|\mathcal{M}(p)|$ problèmes de transport classiques (donc aisément solvables).

$$\min \sum_{o\in\mathcal{O}} \sum_{d\in\mathcal{D}} S_{od}^{m(p)\ell} \cdot w_{od}^{m(p)} \tag{74}$$

sujet à:

$$\sum_{d\in\mathcal{D}} w_{od}^{m(p)} = O_o^{m(p)}, \qquad \forall o \in \mathcal{O} \tag{75}$$

$$\sum_{o\in\mathcal{O}} w_{od}^{m(p)} = D_d^{m(p)}, \qquad \forall d \in \mathcal{D} \tag{76}$$

$$w_{od}^{m(p)} \geq 0, \qquad \forall o \in \mathcal{O}, \forall d \in \mathcal{D} \tag{77}$$

Il devient donc facile d'obtenir toutes les matrices $w^{m(p)\ell*}$, et par une affectation tout-ou-rien, de trouver le flot $z^{p\ell*}$ minimisant le problème approximé.

Nous n'avons, en fait, qu'une simple série de problèmes de transport à résoudre; mais au lieu d'en avoir un par produit comme dans la section précédente, nous en avons un par regroupement de modes. Et une fois de plus, le processus de résolution utilise le même type de procédures que le problème de base, sans différence notoire.

Chapitre V Implantation pratique

Pour compléter l'analyse du problème et de l'approche de résolution développée dans les chapitres précédents nous avons réalisé l'implantation informatique d'un des algorithmes, et testé le comportement de ce programme sur des données réelles. Dans le présent chapitre, nous nous attarderons aux outils dont nous disposons et qui ont dicté le choix de la version de l'algorithme implanté et de la structure du programme. Nous passerons également en revue l'organisation du module et de l'environnement informatique. Enfin, nous donnerons de plus amples détails sur les points importants de ce dernier. Le lecteur intéressé pourra consulter en annexe A les listes complètes des programmes, et trouvera en annexe B une description sommaire des modules de STAN.

1. Outils et choix d'algorithme

1.1. Outils disponibles

L'environnement informatique dont nous disposons est le système d'exploitation UNIX sur ordinateurs SUN, environnement en lui-même extrêmement riche en possibilités et en utilitaires standards. En particulier, nous disposons

des compilateurs des langages suivants: FORTRAN, PASCAL et C.

D'autre part, nous avons accès à plusieurs librairies de programmes et logiciels disponibles au Centre de recherche sur les transports, notamment le progiciel STAN [3] d'analyse stratégique des réseaux de transport et la librairie de procédures RNET [14] pour la résolution de problèmes de transbordement et de transport par la méthode du simplexe révisé.

A l'exception de la résolution d'un problème de transport, le progiciel STAN dispose de tous les outils dont nous avons besoin pour implanter la procédure: calcul des chemins les plus courts en terme des coûts marginaux, affectation d'une matrice sur un réseau multimodal, calcul du pas optimal et de la valeur du *gap*, manipulation de matrices, extraction de données et impression de résultats. De plus, chaque module de ce progiciel peut être appelé de façon individuelle en mode de traitement par lots.

RNET, d'autre part, est une librairie de procédures écrites en FORTRAN, dont les résultats et les performances sont connus depuis longtemps. Elle doit être appelée et contrôlée par un programme principal, lequel fournit le réseau et extrait les résultats.

1.2. Problématique de choix

Le choix d'approche de programmation est donc le suivant:

- soit créer un programme entier qui reproduise l'intégralité de l'algorithme dans un langage classique tel PASCAL, FORTRAN, C, ...
- soit profiter des nombreuses possibilités et de la souplesse des outils déjà présents pour en tirer un maximum d'informations et de services.

C'est pour cette dernière solution que nous avons opté, notre but n'étant

pas de développer une nouvelle application, mais seulement de tester, de la façon la plus complète possible, le comportement de l'algorithme.

1.3. Algorithme implanté

Une conséquence directe de notre choix de programmation est la restriction de l'implantation à l'algorithme B (l'affectation d'un produit unique à un réseau multimodal). Il n'est pas possible en effet, dans la version actuelle de STAN, de calculer des coûts comme ceux requis pour l'application multiproduit.

Les données réelles de Finlande sur lesquelles s'effectuent les tests motivent également ce choix. Elles concernent en effet la distribution d'un seul produit: le charbon.

2. Structure et organisation

2.1. Structure du programme

L'algorithme proposé dans le présent mémoire est aussi un outil d'analyse stratégique des réseaux dans un contexte qu'il n'est pas rare de trouver dans la réalité. Il peut donc s'inscrire naturellement comme module d'affectation alternatif dans le progiciel STAN, module pour lequel la matrice d'affectation de départ serait inconnue ou arbitraire. Cette constatation est la motivation principale qui nous a poussée à reprendre fidèlement tous les formats standards de dialogue utilisés dans STAN, et surtout à inclure notre programme dans la librairie en lui associant le numéro **5.51**.

Dans cette même optique, et pour conserver toute la souplesse de l'outil, la génération d'une matrice initiale a, de plus, été physiquement séparée de

l'algorithme proprement dit pour laisser à l'usager, s'il le désire, la possibilité de désigner une matrice de départ arbitraire.

En résumé, le module **5.51** propose à l'usager les options suivantes:
- créer une matrice de départ par la méthode de balancement (algorithme D),
- exécuter une affectation par l'algorithme B.

2.2. Organisation générale

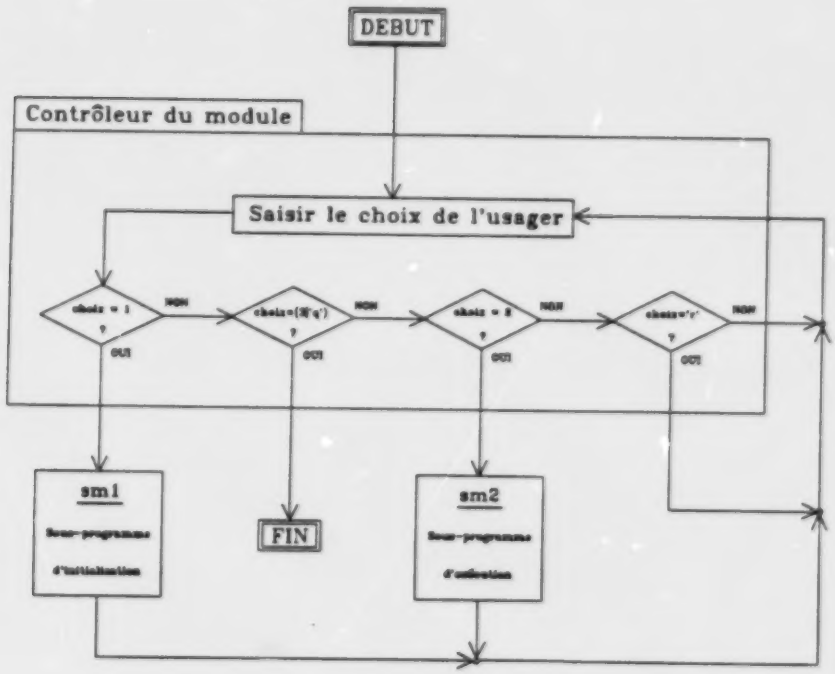


Figure 5: Organisation du module **5.51**.

La figure 5 ci-haut résume l'organisation générale du module: les sous-programmes sont séparés et matérialisent les deux options expliquées précé-

demment. Le contrôleur du module est un programme également séparé des deux autres. Il saisit le choix de l'usager, et en fonction de ce choix, appelle l'un ou l'autre des sous-programmes.

Chaque sous-programme est lui-même une série d'appels aux modules adéquats de STAN, à certains programmes de la librairie standard du système UNIX et à d'autres programmes résidents dans notre système. L'indépendance totale de ces diverses entités dicte assez naturellement l'emploi du **c-shell** comme langage de programmation, puisqu'il s'agit d'un des langages du système d'exploitation UNIX.

2.3. Dialogues et rapports

Les appels aux modules de STAN se font en mode de traitement par lots. Cela nécessite un fichier de dialogue où se trouvent consignées par anticipation toutes les réponses du dialogue qui aurait lieu si le mode d'appel était interactif.

Dans la perspective de notre application, nous pouvons distinguer trois types de tels dialogues:

- ceux qui ne varient pas d'une exécution à l'autre du programme,
- ceux qui ne varient pas d'une itération à l'autre au cours d'une même exécution,
- ceux qui varient d'une itération à l'autre au cours d'une même exécution du programme.

La présence de ce dernier type exclut le concept de fichiers permanents, précrées et domiciliés quelque part dans le système informatique. Aussi devons-nous créer l'ensemble de ces fichiers de dialogue lors d'une étape d'initialisation.

Enfin, les divers modules de STAN génèrent des rapports dont le contenu est d'une grande utilité pour la gestion des appels successifs et des diverses procédures de contrôle de validité. Cependant, l'intégralité de ces rapports n'est pas nécessaire au rapport final de notre programme. Il est donc utile de trier cette information au fur et à mesure qu'elle se présente, et de n'en retenir que l'essentiel dans un fichier temporaire.

2.4. Environnement de travail

Il est important de mettre en relief les divers environnements dans lesquels s'effectue l'exécution du module, et leurs relations respectives. A cet effet, nous en distinguerons trois: deux sont fixes, et le troisième temporaire. La figure 6 résume l'organisation logique du tout.

Les environnements fixes sont les suivants:

- celui de la banque de données sur laquelle s'effectue le travail.
- celui du progiciel STAN où se trouve le module **5.51**.

L'environnement temporaire est créé à chaque appel au module **5.51**, et comprend les fichiers pour les dialogues subséquents de la passe d'exécution.

3. Détails pratiques de l'implantation

3.1. Sous-programme 1 : Création d'une matrice initiale

Créé pour faciliter la tâche à l'usager, ce sous-programme se contente de lui demander les noms de la matrice des sommes de l'offre aux origines, de la matrice des sommes de la demande aux destinations, et de la matrice à créer. Après validation des données, le calcul des chemins les plus courts

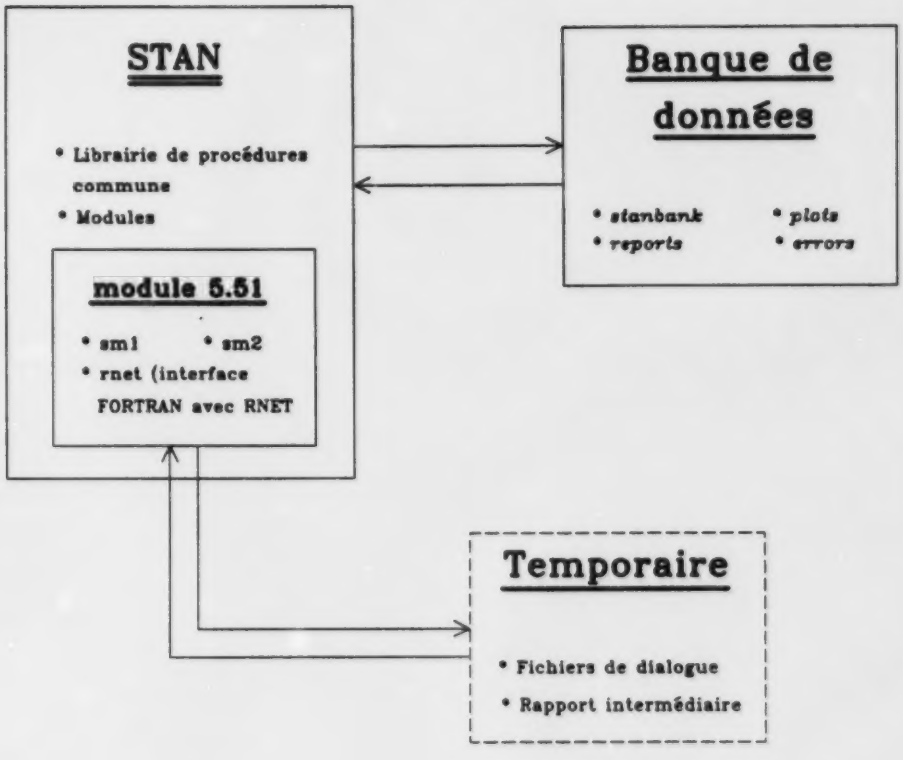


Figure 6: Environnements de travail.

entre centroïdes, le calcul de la réciproque de la matrice de ces chemins et le balancement du résultat ainsi obtenu sont effectués par trois appels successifs aux modules **2.33**, **3.25** et **3.26** de STAN respectivement.

3.2. Sous-programme 2 : Exécution de l'algorithme

La figure 7 montre le schéma du flot d'information dans ce programme. On y distingue trois parties ou étapes que nous allons brièvement décrire et commenter.

a- L'étape préliminaire

Dans cette étape, le programme extrait de la banque de données, grâce au module **1.11** de STAN, les informations nécessaires au contrôle du processus d'optimisation pour le scénario courant, à savoir:

- L'octet de statut du scénario (STATUT). En effet, pour exécuter une affectation par la procédure de Frank et Wolfe adaptée, il faut qu'au préalable le scénario ait été préparé avec le module **5.11**. C'est-à-dire, ou bien l'usager précise quelle matrice de demande affecter au réseau, pour quel produit et selon quel mode; ou bien il spécifie combien d'itérations additionnelles il faut effectuer sur une affectation antérieure.
- Le numéro de l'itération courante (ITER).
- Le nombre maximal d'itérations désirés par l'usager (NITER).

Si le scénario n'est pas prêt pour une affectation, la procédure s'arrête avec un message d'erreur à l'intention de l'usager.

Sinon, le programme commence par mettre le contrôleur d'optimalité (OPT) à **faux**. Puis il saisit les noms des matrices de l'offre (MATO), de la demande (MATD), et de travail (MATC) (cette dernière servira essentiellement à recueillir les coûts marginaux à chaque itération). Il extrait ensuite les paramètres de l'affectation préparée, en vue de l'étape d'initialisation; il s'agit des noms, de la matrice à affecter (MATOD), du produit affecté (PRODS), et de la liste des modes permis (MODES). Après avoir validé toutes ces informations, il saisit enfin le mode de sortie des résultats désiré par l'usager; c'est le seul moment où ce dernier aura la possibilité de renoncer à l'exécution du module par un ordre de fin prématurée.

b- L'initialisation :

La première opération à effectuer est la création de tous les fichiers de dialogue pour les appels futurs aux modules de STAN. Un résumé de l'affectation

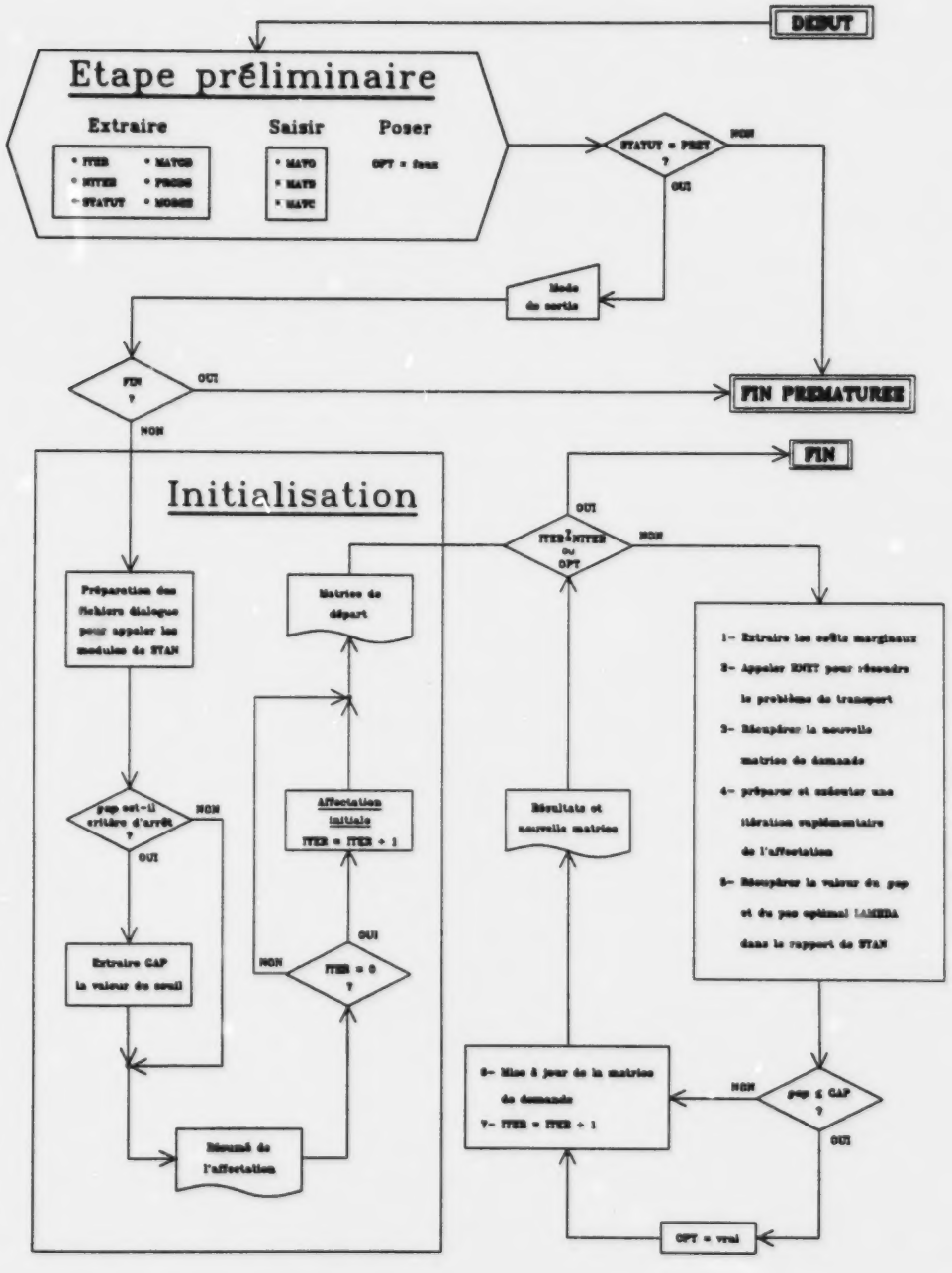


Figure 7: Flot d'information dans le programme **sm2**.

est ensuite obtenu par un appel au module **6.01** de STAN, et inséré dans le rapport d'exécution.

Rappelons ici que notre but est d'effectuer une seule itération à la fois du module **5.21** d'affectation standard; car entre une itération et la suivante, nous voulons pouvoir modifier la matrice de demande en fonction du résultat du problème de transport résolu par RNET, de l'affectation de ce dernier au réseau, et du pas optimal de cette affectation. Nous avons pour cela, encore une fois recours aux services du module **1.11** pour modifier le nombre maximal d'itérations et le ramener à celui de l'itération courante.

Si le problème est absolument nouveau, c'est-à-dire, si l'itération courante est l'itération zéro, une première affectation tout-ou-rien permettra d'obtenir des flots initiaux.

Dans tous les cas, les valeurs de la matrice de demande initiale sont imprimées dans le rapport.

c- La boucle principale :

Deux critères contrôlent l'arrêt du programme:

- l'atteinte du nombre maximal d'itérations requises (ITER = NITER),
- l'atteinte de l'optimalité (OPT = vrai).

Ce dernier cas se produit s'il n'y a pas de recherche linéaire (c'est-à-dire quand les coûts du problème sont linéaires), ou si le *gap* mesuré est inférieur à un certain seuil.

Dans le cœur même de la boucle, le module **6.33** permet d'extraire, dans la matrice (MATC), la valeur des chemins les plus courts en termes des coûts marginaux de la solution courante, tandis que le module **3.14** permet une sauvegarde formatée de cette même matrice dans un fichier physique de l'environnement de la banque de données.

Ce dernier fichier est lu en entrée par un petit programme écrit en FORTRAN qui utilise cette information pour créer la structure de réseau interne, en vue de résoudre le problème de transport par un appel à la procédure RNET. Au retour de cet appel, il récupère, dans la valeur des flots du réseau de transport, la matrice solution du sous-problème linéaire, qui est sauvegardée dans un fichier de l'environnement temporaire du module.

Dans un premier appel au module **3.12**, la matrice actuelle de demande est sauvegardée dans la matrice de travail (MATC ⟵ MATOD) puisqu'elle sera nécessaire pour mettre à jour la demande en fin d'itération.

Dans un second appel, la matrice de demande (MATOD) est modifiée et devient la matrice optimale du problème de transport.

Il ne reste plus qu'à préparer par le module **5.11**, et à effectuer par le module **5.21**, l'affectation de la matrice optimale du problème de transport aux plus courts chemins du réseau en termes des coûts marginaux de la solution courante. On obtient ainsi les flots mis à jour pour la prochaine itération de la boucle principale.

En effet, le module **5.21** effectue la recherche linéaire entre le flot de départ (c'est-à-dire la solution courante), et le flot de la solution extrêmale (c'est-à-dire celle de l'affectation aux plus courts chemins de la matrice résultant de la résolution du problème de transport), et donne ainsi comme résultat le flot déjà mis à jour.

Comme la recherche du pas optimal, de même que le calcul du *gap*, est intégrée au module **5.21**, l'analyse de l'état d'optimalité de la solution actuelle se fait par simple recherche à travers le fichier *reports* généré par STAN. Cette recherche peut se faire de façon efficace grâce, en particulier, aux deux utilitaires UNIX **grep** et **awk**. La valeur du pas optimal et celle

du *gap* sont récupérées et analysées, et le contrôleur d'optimalité modifié, le cas échéant.

Enfin, avec le module **3.25** la matrice de demande (MATOD) est mise à jour comme à l'étape 5 de l'algorithme B.

Il ne reste plus qu'à incrémenter le compteur d'itérations et à imprimer le rapport de l'itération courante selon le mode choisi par l'usager, sur l'écran, ou à la fin du fichier de rapport.

3.3. Limitations du module expérimental

Pour diverses raisons, que nous présenterons au fur et à mesure, il ne nous a pas été possible d'implanter l'algorithme B avec le degré de généralité voulu. Aussi, notre programme présente-t-il un certain nombre de limitations.

a- Vérification des données de l'usager

Nous avons fourni un maximun d'effort pour vérifier et tester, chaque fois qu'il l'était possible et raisonnable, la cohérence des informations données par l'usager. En particulier, l'existence préalable de toutes les matrices, y compris la matrice de demande initiale à calculer dans le sous-programme 1, est strictement requise. Par contre, tant pour le sous-programme 1 que pour le sous-programme 2, aucune vérification n'est faite sur la correspondance des matrices données et l'on présuppose que l'usager s'en sera chargé.

Autrement dit, il est de sa responsabilité de s'assurer que les matrices contenant les totaux de l'offre (mo) et les totaux de la demande (md) dont il fournit les noms, contiennent bien toutes les informations nécessaires au sous-programme 1, et que la matrice de demande (mf) ne contient aucune information dont la perte soit irrémédiable.

De même, dans le sous-programme 2, il est également de sa responsabilité de s'assurer que les éléments de la matrice de l'offre correspondent bien aux totaux sur les lignes de la matrice de demande initiale (mf) qu'il a fournie, et que les éléments de celle de la demande correspondent bien aux totaux sur les colonnes de la matrice (mf).

b- Format de saisie

Il est malaisé de faire des vérifications sur les chaînes de caractères en **c-shell**. Comme dans bien des situations il est illusoire et inutile de vouloir prévoir tous les cas possibles, nous avons été conduits à poser un format fixe pour les noms de matrices, contrairement à la procédure en vigueur dans la saisie de données de STAN.

En effet, l'usager de STAN peut désigner une matrice, disons mf3, indifféremment par: **mf3** ou **mf03**, le programme se chargeant de faire la conversion. Il n'en est pas de même dans notre programme, qui s'appuie, pour vérifier l'existence d'une matrice, sur l'analyse de la liste des matrices existantes créées par le module **3.14**, liste caractérisée par le format fixe suivant: $m\alpha\#\#$, où: $\alpha \in \{o, d, f\}$ et $\# \in [0, 9] \bigcap \mathrm{N}$.

c- Fragilité

Une grande partie de la gestion des sous-programmes se fonde sur l'analyse du fichier *reports* généré par STAN lors de l'appel à ses diverses fonctions. Elle se base donc fortement sur l'existence et le format d'écriture de mots-clefs comme "Gap:" ou "L*" pour récupérer l'information nécessaire au travail et se retrouve ainsi extrêmement dépendante de ce format; elle est par conséquent fragile.

Chapitre VI Résultats numériques

Nous voici en possession d'un modèle mathématique, d'une approche de résolution de ce modèle, et d'un programme matérialisant l'algorithme de résolution. Il ne nous reste donc logiquement plus qu'à tester le comportement de cet algorithme dans différents cas de figure. Pour effectuer ces tests, nous disposons, comme déjà mentionné, de données réelles relatives à l'importation et la distribution du charbon en Finlande.

Dans le présent chapitre, nous présenterons brièvement le réseau de distribution finlandais avec toutes ses caractéristiques initiales; nous présenterons aussi les différents tests qui ont été envisagés pour l'analyse; nous résumerons enfin les résultats obtenus avant de donner quelques conclusions qu'ils nous suggèrent quant à la pertinence et à l'utilité de cette approche.

1. Les caractéristiques du réseau finlandais

1.1. Le réseau de test

Le réseau finlandais est composé de 2539 nœuds, 33 centroïdes, 6648 liens et 1772 transferts. Un seul produit y est transporté: le charbon, et à chaque mode de transport est associée une fonction de coût unique. Tous les cen-

troïdes d'offre sont situés outremer, alors que tous les centroïdes de demande sont nationaux; cela implique un passage forcé par l'un ou l'autre des modes maritimes[2], mais aucun transfert n'est permis entre les divers modes maritimes sauf à l'embouchure de l'unique voie fluviale navigable, aux confins de l'URSS et de la Finlande. Enfin, le coût (en marks finlandais) de transport du charbon a la structure générale suivante:

$$\begin{aligned} \text{Coût total} &= \text{Coût de transport pur} \\ &+ \text{Coût fixe} \\ &+ \text{Coût de transbordement pur} \end{aligned}$$

Le coût unitaire de transport pur s'applique à la tonne-kilomètre, tandis que l'unité pour le coût fixe et celui de transbordement pur est la tonne. Le coût fixe est payé chaque fois qu'un nouveau mode est utilisé, donc chaque fois qu'il y a transbordement de charbon. On peut lui donner l'interprétation de frais d'amortissement du matériel de transport et autres frais connexes à ceux de transbordement pur.

Le tableau II ci-dessous résume l'ensemble des caractéristiques physiques du réseau.

Nœuds	Nombre	Arcs	Nombre
Simple	2332	Liens	6648
Transfert possible	207	Transferts	1772
Centroïdes d'offre	10		
Centroïdes de demande	23		
Total	2572	**Total**	8420

Tableau II: Caractéristiques physiques du réseau finlandais.

[2]Voir à section suivante le détail de la modélisation des modes de transport

1.2. Les modes

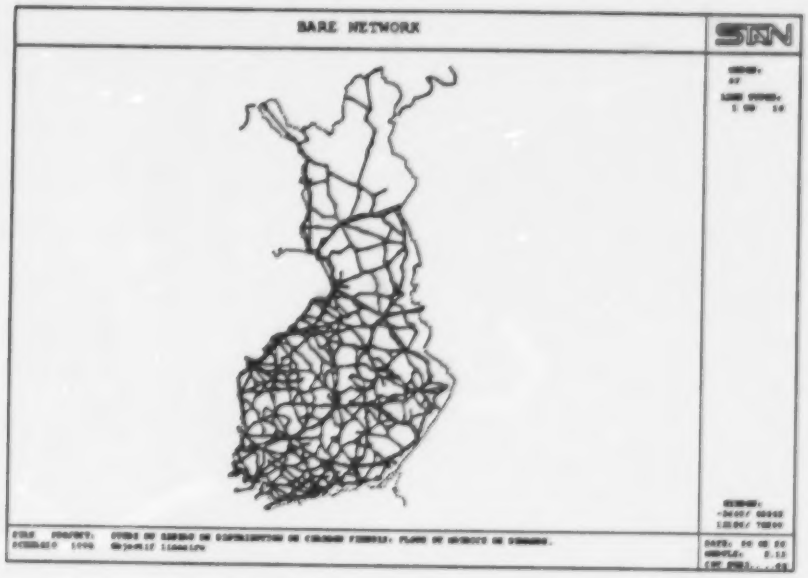
De prime abord, le réseau finlandais est un réseau trimodal: le charbon peut être distribué par les voies navigables, le chemin de fer ou la route. L'étude approfondie de la structure de coût montre cependant, d'une part, que le nombre de modes est en fait plus élevé en raison de la diversité des types de voies navigables et des contraintes de tirant d'eau qui leur sont associées; et de l'autre, que le contrôle de l'accès aux centroïdes et l'isolation des coûts de transbordement suggèrent l'isolement des liens de connexion grâce à un mode fictif.

En effet, les voies navigables profondes peuvent, non seulement accueillir les gros cargos, mais aussi les embarcations de plus faible tonnage. Par contre, les voies peu profondes ne sont accessibles qu'aux embarcations ayant un tirant d'eau inférieur à une certaine valeur critique.

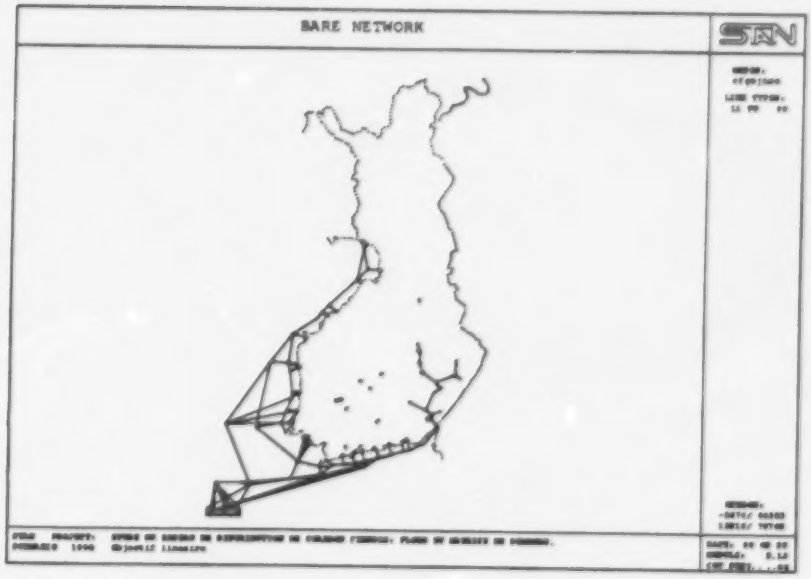
Or les données dont nous disposons identifient sept types différents de voies navigables, comme le montre le tableau III ci-dessous.

Type	Description du type de voies
1	Tirant d'eau inférieur à 5 mètres (écluses)
2	Tirant d'eau inférieur à 7 mètres (rivières)
3	Tirant d'eau inférieur à 9 mètres (voies maritimes)
4	Tirant d'eau inférieur à 10 mètres (*ib.*)
5	Tirant d'eau inférieur à 13 mètres (*ib.*)
6	Tirant d'eau inférieur à 15.3 mètres (*ib.*)
7	Tirant d'eau supérieur ou égal à 15.3 mètres (*ib.*)

Tableau III: Réseau navigable finlandais.



a- Réseau routier et ferroviaire de Finlande.



b- Réseau navigable de connexion.

Figure 8: Le réseau de transport finlandais.

De plus, à ces différents types de voies sont associées des fonctions de coût qui s'appliquent aux embarcations de la plus grande taille accessible à la voie considérée. Nous pourrions envisager de mettre en relief ces différences dans le coût d'exploitation au niveau de la fonction associée à chaque lien navigable. Malheureusement, cette approche présuppose que les navires de la plus grande taille possible seront seuls utilisés sur une voie maritime donnée, ce qui ne correspond certainement pas à la réalité, et pas nécessairement non plus à la solution de moindre coût. Outre cet inconvénient, un chemin optimal entre une origine o, et une destination d pourrait alors comporter, sur la partie navigable du parcours, des liens dont les fonctions seraient différentes; ce qui impliquerait la nécessité de transbordements en haute mer entre embarcations de tonnage différents afin de minimiser le coût réel d'exploitation, or ceci ne correspond en soi à aucune réalité, et, de toutes façon, ferait intervenir un coût supplémentaire qui n'avait pas été pris en compte.

Il paraît donc raisonnable d'identifier chaque "classe" de bateaux à un mode différent. Cela peut se déduire assez facilement du tableau III. Ainsi, une voie navigable sera représentée par un lien dans le réseau de chaque mode qui y a accès et les transbordements n'auront lieu qu'aux seuls endroits identifiés à cette fin par des liens de transfert.

Les centroïdes sont reliés au réseau par des liens fictifs dont l'attribut mode correspond à un mode de transport fictif. Cela facilite en effet le contrôle, grâce à la fonction de coût de transfert, des types de véhicules habilités à charger ou décharger. Il suffit en effet, de mettre un coût unitaire "infini" sur un lien de transfert pour être assuré qu'il ne sera pas utilisé. De plus, il est ainsi plus aisé d'isoler les coûts de transbordement pur dans le réseau pour fin d'analyse.

Symbole	Description du mode	Tirant d'eau (t.e.), en mètre, des embarcations	Nombre de liens	Longueur totale* (km)
a	Véhicules routiers		5422	39543
r	Chemin de fer		366	11627
c	Connecteurs des centroïdes		68	0
f	Barges:	t.e. < 05.0	20	4
g	Embarcations:	05.0 ≤ t.e. < 07.0	56	1291
h	Embarcations:	07.0 ≤ t.e. < 09.0	180	114476 (334)
j	Embarcations:	09.0 ≤ t.e. < 10.0	166	114142 (630)
k	Embarcations:	10.0 ≤ t.e. < 13.0	152	113512 (3688)
m	Embarcations:	13.0 ≤ t.e. < 15.3	114	109824 (226)
n	Embarcations:	15.3 ≤ t.e.	104	109598

Tableau IV: Description des modes du réseau finlandais.

* Comme tous le réseau est bidirectionnel, mais que les arcs sont unidirectionnels, la longueur totale doit en fait être divisée par deux pour avoir les dimensions réelles. Les chiffres entre parenthèses indiquent la longueur cumulative totale des voies navigables inaccessible à un tirant d'eau supérieur. Par exemple, 226 parmi les 109598 km de voies navigables avec des embarcations de type m ne sont pas assez profondes pour celle de type n.

2. Les scénarios des tests[3]

Les tests dont la description suit mettent en œuvre trois algorithmes que nous noterons **ACM**, **AMF**, et **HEU**. Ces sigles correspondent respectivement aux noms suivants:

- Affectation avec Calcul de Matrice: il s'agit de l'algorithme B développé au chapitre II.
- Affectation avec Matrice Fixe: il s'agit de l'algorithme d'affectation implanté dans STAN (voir [15]).
- Heuristique: il s'agit d'une méthode heuristique que nous présenterons plus loin.

Sur le réseau décrit à la section précédente, nous avons mis en œuvre trois groupes de tests, correspondant chacun à une structure de coût particulière; nous les désignerons par groupe I, II et III.

Tous les tests du groupe I ont été effectués sur les données d'origine, c'est-à-dire sur le réseau finlandais tel que déjà décrit, et avec la structure de coût fournie par les données de ce réseau. Or, toutes les fonctions de coût dont nous disposions étaient linéaires.

Les deux premiers tests de ce groupe sont une application simple de notre algorithme, qui mettent en relief la non-unicité de la matrice de demande optimale, le deuxième test ayant été réalisé en modifiant légèrement l'interface avec la librairie RNET de façon à créer une permutation sur la numérotation des arcs et des nœuds dans le réseau interne de RNET. Ces deux premiers tests mettent en œuvre l'ACM.

Un troisième test a consisté en une résolution du problème par la méthode

[3]Dans la suite de ce chapitre, "fonction de coût" ou "coût" désigneront la fonction de coût total ou le coût total respectivement sauf quand il sera spécifié autrement.

heuristique suivante (mat$n$ est une matrice et cpc les chemins les plus courts):

0. Calculer mat1 avec l'algorithme D
1. Faire l'AMF de mat1 au réseau
2. Extraire mat2: les cpc en termes des coûts marginaux
3. Calculer mat3 = exp(–0.01 × mat2)
4. Balancer mat3 avec l'algorithme D
5. mat1 ⟵ mat3 et retourner en 1.

HEU: Méthode heuristique de résolution.

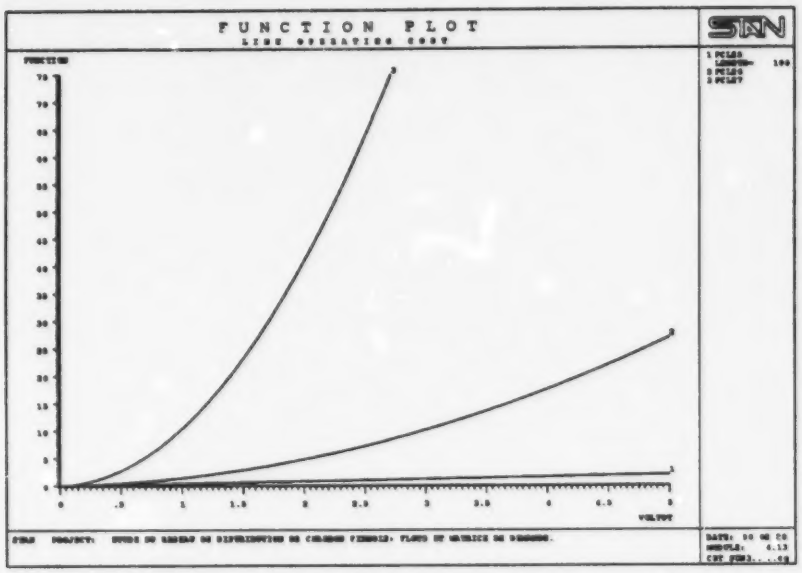
Les groupes de tests II et III ont été effectués, d'une part sur le réseau d'origine, et d'autre part sur une version légèrement modifiée de ce réseau, dans laquelle nous avons imposé à certains liens une capacité de flot maximale en modifiant convenablement les fonctions de coût.

Pour les tests du groupe II nous avons considéré des fonctions de coût, soit linéaires, soit légèrement non-linéaires. Ces dernières ont été créées en introduisant dans les fonctions d'origine un terme quadratique à faible coefficient. Pour les véhicules routiers, par exemple, la fonction de coût total de l'arc a est définie comme suit:

$$s_a(v_a) * v_a = ((0.27 * long_a) + (0.01 * voltot_a)) * voltot_a$$

où $long_a$ désigne la longueur de l'arc a, mesurée en kilomètres, et $voltot_a$, le volume total de flot traversant l'arc a, mesuré en kilotonnes:

Dans les tests du groupe III nous avons considéré des fonctions de coût linéaires ou fortement non-linéaires. Pour créer ces dernières, nous avons suivi le même principe que pour le groupe II, mais en augmentant le coefficient du terme quadratique de façon significative en le multipliant par la longueur du lien; ce qui a, de plus, rendu les fonctions de coût très variables d'un arc



Fonctions de coûts: [1] linéaire, [2] légèrement non-linéaire et [3] fortement non-linéaire.

Figure 9: Exemples de fonctions de coûts.

à l'autre puisque les longueurs varient d'une fraction de kilomètre au millier de kilomètres. Le mode maritime **n**, par exemple, (voir tableau IV) présente la fonction de coût moyen suivante:

$$s_a(v_a) = (0.004 * long_a) + (0.01 * voltot_a * long_a)$$

Pour chacun des groupes II et III ainsi défini, et pour chaque réseau, nous avons effectué une série de trois affectations:

- Un premier test a consisté à appliquer l'ACM.
- Dans un second test, nous avons effectué l'AMF en utilisant la matrice initiale du premier test.
- Dans un dernier test, nous avons effectué également une AMF, mais en utilisant cette fois-ci la dernière matrice obtenue par le premier.

Le tableau V ci-après résume les caractéristiques de l'ensemble des tests:

Groupe	Fonctions de coût	Réseau	Test	Algo-rithme	Description	Nombre d'itérations
I	Linéaires	Origine,	1000	ACM		1
		sans	4000	ACM	Modification de l'interface RNET	1
		capacités	7000	HEU	Méthode heuristique de résolution	3
II	Linéaires	Origine,	2000	ACM		10
	et faiblement	sans	2200	AMF	Affectation de la matrice initiale du test 2000	10
	non-linéaires	capacités	2500	AMF	Affectation de la matrice finale du test 2000	10
		Modifié,	3000	ACM		10
		avec	3200	AMF	Affectation de la matrice initiale du test 3000	10
		capacités	3500	AMF	Affectation de la matrice finale du test 3000	10
III	Linéaires	Origine,	5000	ACM		30
	et fortement	sans	5200	AMF	Affectation de la matrice initiale du test 5000	30
	non-linéaires	capacités	5500	AMF	Affectation de la matrice finale du test 5000	30
		Modifié,	6000	ACM		30
		avec	6200	AMF	Affectation de la matrice initiale du test 6000	30
		capacités	6500	AMF	Affectation de la matrice finale du test 6000	30

Tableau V: Description des divers tests numériques.

3. Résultats des tests

Les résultats les plus intéressants sont les comportements propres à chacun des groupes de test. Avant d'élaborer davantage sur la question, faisons au préalable une remarque rapide:

Le comportement du *gap* en fonction du nombre d'itérations ne semble absolument pas influencé par la variabilité de la matrice (voir le tableau VI et la figure 10).

Test	*gap*		Nombre	Amélioration
	début	fin	d'itérations	Globale
1000 (4000)	0.00%	0.00%	1	7.11%
7000	0.00%	0.00%	3	0.32%
2000	7.61%	0.21%	10	7.02%
2200	7.61%	0.24%	10	1.74%
2500	7.85%	0.21%	10	1.32%
3000	16.83%	0.59%	10	9.66%
3200	16.83%	0.39%	10	4.71%
3500	7.72%	0.17%	10	1.31%
5000	100.00%	4.14%	30	62.40%
5200	100.00%	4.89%	30	62.15%
5500	100.00%	4.33%	30	61.56%
6000	100.00%	6.14%	30	67.28%
6200	100.00%	6.81%	30	66.95%
6500	100.00%	6.02%	30	66.89%

Tableau VI: Résultats par test.

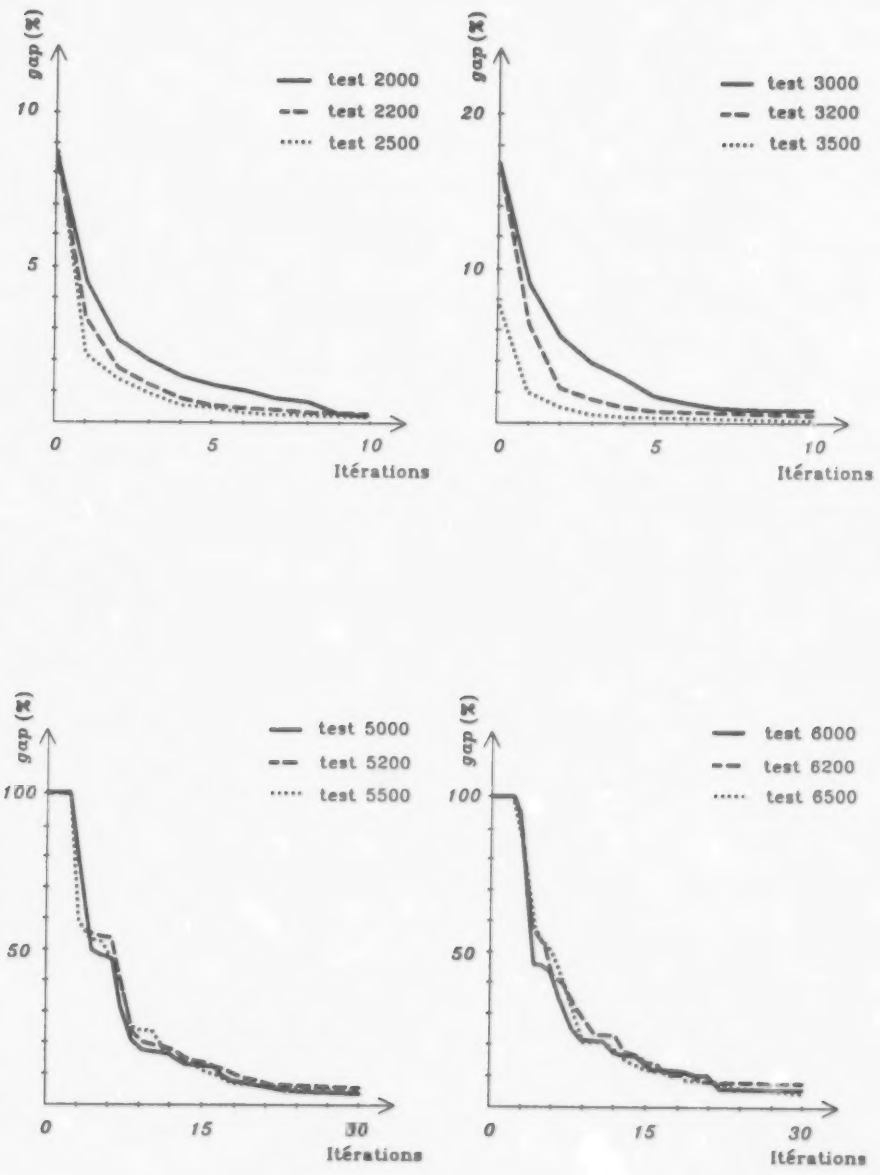


Figure 10: Comportement du *gap* en fonction du nombre d'itérations.

3.1. Groupe I: fonctions linéaires

Comme prévu, l'algorithme B s'arrête après la première itération de l'étape générale. L'amélioration de l'objectif du problème par l'ACM est de l'ordre de 7.11%, alors que l'AMF s'arrête immédiatement après l'initialisation. La tentative de résolution par la méthode heuristique s'est avérée quant à elle tout à fait infructueuse, puisque la matrice a cessé "d'évoluer" au bout de la première itération ne donnant qu'une amélioration de 0.32% de la valeur de l'objectif.

3.2. Groupe II: fonctions linéaires et faiblement non-linéaires

Trois constatations se dégagent des résultats de ces tests:

- L'amélioration globale de la valeur de l'objectif obtenue grâce à l'ACM est significativement plus importante que celle obtenue grâce à l'AMF.
- L'application de l'AMF avec, pour matrice de demande, la matrice "optimale" obtenue grâce à l'ACM, ne donne pas de résultats significativement meilleurs au bout du même nombre d'itérations.
- Pour une matrice de départ "quelconque" (arbitraire) le résultat de l'AMF peut en revanche être très mauvais.

3.3. Groupe III: fonctions linéaires et fortement non-linéaires

Ce qui frappe dans les résultats de ces tests, c'est l'identité de comportement entre l'ACM et l'AMF indépendamment de toute matrice: même taux énorme d'amélioration de la valeur de l'objectif, même cheminement, ou

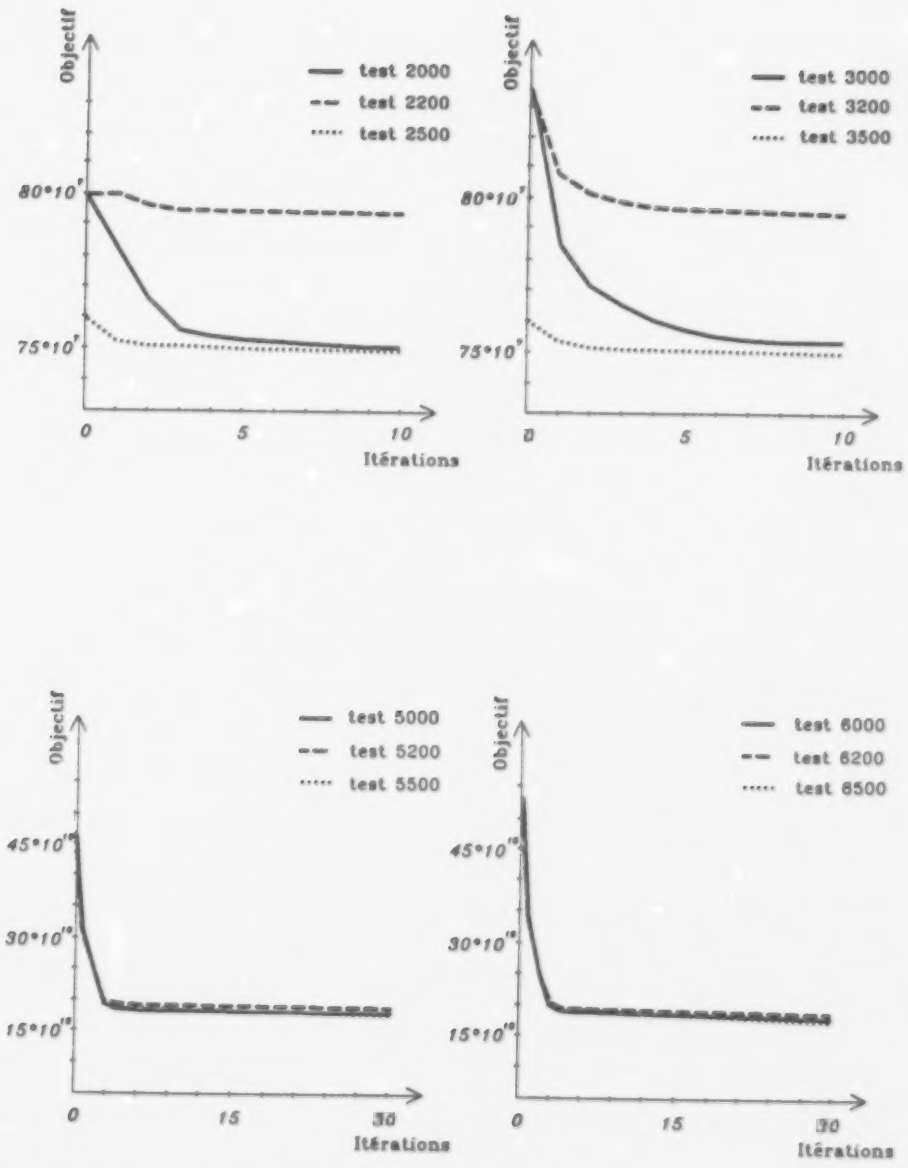


Figure 11: Comportement de l'objectif en fonction du nombre d'itérations.

presque, vers l'optimum, puisque la différence relative entre les valeurs de l'objectif à itérations correspondantes n'est pas significative, quelle que soit la paire de tests considérés dans ce groupe (voir tableau VII).

Tests comparés	Différence relative* de l'objectif		
	après l'initialisation	après l'itération	numéro
2000 – 2200	0.00%	-5.67%	10
2000 – 2500	5.82%	0.04%	10
2200 – 2500	5.82%	5.41%	10
3000 – 3200	0.00%	-5.48%	10
3000 – 3500	8.88%	0.46%	10
3200 – 3500	8.88%	5.63%	10
5000 – 5200	0.00%	-0.64%	30
5000 – 5500	2.31%	0.12%	30
5200 – 5500	2.31%	0.76%	30
6000 – 6200	0.00%	-0.99%	30
6000 – 6500	1.05%	-0.11%	30
6200 – 6500	1.05%	0.87%	30

* Le signe (-) devant la différence relative indique que l'objectif du test de gauche est inférieur (meilleur) que celui du test de droite.

Tableau VII: Comparaison des résultats des tests.

4. Commentaires

L'ensemble de ces tests met clairement en relief l'intérêt de la méthode de l'ACM par rapport à l'AMF lorsque, d'une part, la structure de coût est linéaire ou faiblement non-linéaire, et que de l'autre, la définition de la ma-

trice de demande n'est soumise à aucune contrainte et ne dépend que de l'usager. Dans de tels cas, même si ce dernier disposait de la matrice "optimale" — ce qui reste encore à prouver en l'absence de tout critère permettant de définir *a priori* de telles matrices — l'application de l'ACM reste avantageuse, puisqu'elle permet, une économie substantielle au niveau des coûts de transport, et que cette économie se manifeste également au niveau du travail nécessaire pour obtenir une matrice "optimale" par des moyens empiriques ou mathématiques — travail qui serait du reste à reprendre entièrement dès qu'interviendrait la moindre variation dans la structure de coût du réseau.

Il en ressort aussi très clairement que l'ACM n'offre par contre aucun intérêt quand la structure de coût du réseau est trop non-linéaire ou complexe. L'ACM recourt déjà en effet à l'AMF pour l'étape ℓ-1 de l'algorithme B dans chaque itération de l'étape générale; elle nécessite en outre un algorithme de programmation linéaire, sans compter de nombreuses manipulations de matrices ce qui entraîne une structure bien lourde pour fort peu de résultats.

Conclusion

Pour terminer cette étude, soulignons encore une fois l'intérêt de notre approche, avant d'indiquer quelques avenues de recherche qui se dégagent au terme de ce travail.

Rappelons les points essentiels du problème. Nous désirons trouver à la fois la matrice de demande optimale d'un réseau de distribution et les flots de coût minimum correspondants, sachant que:

1- La matrice optimale n'est pas unique.

2- Il n'existe pas de critères permettant d'estimer à l'avance l'optimalité d'une matrice.

3- Nous sommes dans un contexte où l'opérateur a le contrôle total sur la distribution des flots: les flots observés ne traduisent donc pas une réalité inéluctable dont on ne saurait s'éloigner dans la recherche de meilleurs résultats.

Nous avons donc conçu et testé une méthode itérative de diminution de la valeur de l'objectif, combinant la recherche des flots à celle de la matrice de demande. L'implantation de cette méthode dans le cas du transport d'un produit unique s'avère substantiellement intéressante quand la structure des coûts sur le réseau est proche de la linéarité. Rappelons ici que l'application aux données brutes de Finlande a permis une économie de 7.11% sur des

coûts qui sont de l'ordre du demi milliard de marks finlandais[4]. Une étude ultérieure pourrait tester le comportement de l'algorithme et en explorer les possibilités dans le cadre de son application multiproduit.

Une première remarque s'impose: l'approche de résolution combinée que nous avons suivie n'est pas la seule possible. En effet, comme mentionné dans la remarque 3 ci-haut, les flots sont ceux que le gestionnaire décide qu'ils soient. Supposons que nous disposions des flots optimaux, il suffirait alors d'appliquer une méthode numérique d'extraction de la matrice de demande (comme, par exemple, la méthode de décomposition des flots en flots de chemins) pour obtenir une des matrices correspondant à la solution optimale. Il s'agirait donc de résoudre un problème de flot à coût minimal entre les origines et les destinations en vue d'obtenir le flot optimal, et, partant, de trouver une matrice correspondante. L'exploration d'approches alternatives de ce type permettrait d'établir clairement les données de l'arbitrage qui s'impose quant au choix de l'algorithme de résolution du problème de base: d'une part, en effet, nous devons traîner en permanence une matrice additionnelle pour recueillir les résultats du sous-problème linéaire, ce qui réduit considérablement la taille des problèmes pouvant être résolus en pratique; et de l'autre, l'effort et le temps-machine nécessaires à la décomposition des flots peuvent coûter extrêmement cher selon la taille du problème.

Une seconde remarque est relative aux difficultés 1 et 2, qui mettent en relief l'utilité d'établir des critères et une méthodologie permettant de classifier, voire même de générer des, ou toutes les matrices optimales d'un problème. Mais, il suffit de se rendre compte qu'un léger changement dans la structure de coût du réseau se répercute gravement sur l'optimalité d'une matrice, pour constater l'impossibilité de mener une étude aussi générale.

---

[4] 1 MF ≈ 0.3 $C

Bibliographie

[1] Assad, A. A., *Multicommodity Network Flows — A Survey*, Networks, Vol. 8, pp. 37–91, 1979.

[2] Andersson, P. A., *On the Convergence of Iterative Methods for the Distribution Balancing Problem*, Transportation Research, Vol. 15B, pp. 173–201, 1981.

[3] Crainic, T. G., Florian, M., Guélat, J., Larin, D., *STAN User's Manual*, Centre de recherche sur les transports, Université de Montréal, 1988.

[4] Crainic, T. G., Florian, M., Guélat, J., Spiess, H., *Strategic Planning of Freight Transportation: STAN, an Interactive-Graphic System*, Publication #664, Centre de recherche sur les transports, Université de Montréal, 1989, à paraître dans Transportation Research Record.

[5] Erlander, S., *Accessibility, Entropy, and the Distribution and Assignment of Traffic*, Transportation Research, Vol. 11, pp. 149–153, 1977.

[6] Erlander, S., Nguyen, S., Stewart, N., *On the Calibration of the Combined Distribution-Assignment Model*, Transportation Research, Vol. 13B, pp. 259–267, 1979.

[7] Evans, S. P., *Derivation and Analysis of Some Models for Combining*

Trip Distribution and Assignment, Transportation Research, Vol. 10, pp. 37–57, 1976.

[8] Florian, M., *An Introduction to Network Models Used in Transportation Planning*, dans *Transportation Planning Models*, ed. M. Florian, North-Holland Publishing Company, Amsterdam, 1984.

[9] Florian, M., *Nonlinear Cost Network Models in Transportation Analysis*, Mathematical Programming Study, Vol. 26, pp. 167–196, 1986.

[10] Florian, M., Nguyen, S., *A Combined Trip Distribution Modal Split and Trip Assignment Model*, Transportation Research, Vol. 12, pp. 241–246, 1978.

[11] Florian, M., Ferland, J., Nguyen, S., *On the Combined Distribution-Assignment of Traffic*, Transportation Science, Vol. 9, pp. 43–53, 1975.

[12] Frank, M., Wolfe, P., *An Algorithm for Quadratic Programming*, Naval Research Logistics Quarterly, Vol. 3, pp. 95–110, 1956.

[13] Furness, K. P., *Time-Function Iteration*, Traffic Engineering and Control, Vol. 7, pp. 458–460, 1965.

[14] Grigoriadis, M. D., Hsu, T., *RNET — The Rutgers Minimum Cost Network Flow Subroutines*, Rutgers University, New Brunswick, N.J., 1979.

[15] Guélat, J., Florian, M., Crainic, T. G., *A Multimode Multiproduct Network Assignment Model for Strategic Planning of Freight Flows*, Transportation Science, Vol. 24-1 , pp. 25–39, 1990.

[16] Lamond, B., Stewart, N. F., *Bregman's Balancing Method*, Transportation Research, Vol. 15B, pp. 239–248, 1981.

[17] Luenberger, D. G., *Introduction to Linear and Nonlinear Program-*

ming, Addisson-Wesley, Reading, MA, 1984.

[18] Kennington, J. L., *A State-of-the-art Survey of Linear Cost Multicommodity Network Flows*, Operations Research, Vol. 26-2 , pp. 209–236, 1978.

[19] Kennington, J. L., Helgason, R. V., *Algorithms for Network Programming*, Wiley-Interscience, 1980.

[20] Minoux, M., *Programmation Mathématique: Théorie et Algorithmes*, tomes 1 et 2, Dunod, Paris, 1983.

[21] Nguyen, S., *Estimating Origin-Destination Matrices from Observed Flows*, dans *Transportation Planning Models*, ed. M. Florian, North-Holland Publishing Company, Amsterdam, 1984.

[22] Robillard, P., Stewart, N., *Iterative Numerical Methods for Trip Distribution Problems*, Transportation Research, Vol. 8, pp. 575–582, 1974.

[23] Tomlin, J. A., *A Mathematical Programming Model for the Combined Distribution-Assignment of Traffic*, Transportation Science, Vol. 5, pp. 122–140, 1971.

[24] Wardrop, J., *Some Theoretical Aspects of Road Traffic Research*, dans, *Proceedings of the Institute of Civil Engineers*, Part. II, Vol. 1, pp. 325–378, 1952.

[25] Zoutendijk, G., *Mathematical Programming Methods*, North-Holland Publishing Company, Amsterdam, 1976.

Bibliographie thématique

Modèles de réseau et algorithmes de résolution

[8], [9], [19]

Problèmes d'affectation multiproduit: modèles et algorithmes

[1], [15], [18]

Problèmes combinés de distribution-affectation: modèles et algorithmes

[5], [6], [7], [10], [11], [23]

Problèmes de distribution: modèles et algorithmes

[2], [13], [16], [21], [22]

Programmation mathématique

[12], [17], [20], [25]

Références sur STAN et RNET (en sus de la bibliographie de l'annexe B)

[3], [4], [14], [15]

ANNEXE A Listes des Programmes

1. Liste du contrôleur du module s551

```
#!/bin/csh -f
#*****************************************************************************
#         Module "STAN" experimental: Menu de selection principal
#         =======================================================
#
# Ce court programme permet a l'usager de choisir une parmis les quatre options
# suivantes:
#   <1> Creer une matrice de demande a partir des totaux de l'offre et de la
#       demande (ceci est realise par le programme $modenv/sm1).
#   <2> Executer l'algorithme de Frank et Wolfe adapte au probleme d'estimer la
#       matrice de demand (cette execution est controlee par le programme
#       $modenv/sm2).
#   <r> Recommencer la selection.
# <q|3> Quitter le module.
# Une fois que l'usager a fait son choix et que celui-ci a ete valide, la
# procedure provoque l'execution du programme $modenv/sm1 en reponse au choix
# <1>, du programme $modenv/sm2 en reponse au choix <2>, recommence au debut
# si le choix etait <r>, ou termine quand la derniere option est demandee.
#
#*** Variables globales qui seront heritees par les programme sm1 et sm2:
#    ------------------------------------------------------------------
#        stanenv : Repertoire ou se trouve la banque de donnees STAN sur
#                  laquelle va s'effectuer le travail. C'est le repertoire
#                  courant a partir duquel STAN est appele
#        stanmod : Repertoire ou se trouvent les modules de STAN
#        modenv  : Repertoire ou se trouvent les sous-programmes sm1 et sm2
#                  qui gerent les deux options du module
#        repbat  : Repertoire temporaire cree par sm1 et sm2 et ou seront
#                  consignes les resultats intermediaires et les fichiers
#                  de dialogue pour le traitement par lot des autres modules
#                  de STAN
#
#*** Variables locales:
#    ------------------
#        RESTART,FLAG,REP : Variables de travail
#*****************************************************************************

# Initialisation et definition des divers environnement du programme
set RESTART=1
setenv stanenv $cwd
setenv stanmod /usr/crts/christin/finland/finbank/bin
setenv modenv /usr/crts/christin/finland/finbank/bin/s551.sm
setenv repbat /usr/crts/christin/finland/finbank/bin/s551.sm/tmp
cd $stanenv

# Menu principal et boucle de selection
while ( $RESTART )
  echo " 5.51 DEMAND MATRIX ESTIMATION AND ASSIGNMENT"
  echo ""
  echo ""
  echo " Select: Operation"
  echo "         1= initialize a demand matrix"
```

```
    echo "          2= algorithm execution "
    echo "          3= end"
    echo -n " "
    set REP=$<
    set FLAG=1
    while ( $FLAG )
      switch ( $REP )
        case [1] :
          $modenv/sm1
          set FLAG=0
          breaksw
        case [2] :
          $modenv/sm2
          if ( $status == 2 ) exit
          set FLAG=0
          breaksw
        case [r] :
          set FLAG=0
          breaksw
        case [q,3] :
break ; break
          breaksw
        default :
          echo valid answers are 1 2 3 r for restart and q for quit
          set REP=$<
          breaksw
      endsw
  end       # while ($FLAG)
end       # while ($RESTART)

# at eof
```

2. Liste du sous-programme d'initialisation sm1

```
#!/bin/csh -f
#*******************************************************************************
# Programme pour creer une matrice de demande a partir de l'information de la
# valeur totale de l'offre et de la demande aux centroides du reseau.
#
# Notes importantes:
# ------------------
#   --> Les noms de matrice donnes par l'usager doivent correspondre a des
#       matrices deja existantes (c'est a dire initialisees dans la banque
#       de donnees et reconnues par STAN).
#   --> L'usager peut aussi abandonner l'execution du programme en repondant
#       par une ligne vide a l'une ou l'autre des questions
#
#*** Variables globales heritees de la procedure s551:
#    ------------------------------------------------
#       stanenv,stanmod,
#       modenv,repbat    : Repertoires definissant les environnements de travail
#       status           : Octet de statut a la fin d'une execution de programme
#
#*** Variables locales:
#    ------------------
#       REP,FLAG : Variables de travail
#
#       MATOD : matrice de demande a creer
#       MATO  : matrice des totaux disponibles aux origines
#       MATD  : matrice des totaux demandes aux destinations
#*******************************************************************************

# Preparation de l'environnement de travail
  mv $stanenv/reports $stanenv/rapport
  if ( -d $repbat ) then
    rm $repbat/*
  else
    mkdir $repbat
  endif
  if ( -e $repbat/rapport ) rm $repbat/rapport
  cd $stanenv

# Statut de traitement par lots forces pour les appels subsequents
  if ( -e $stanenv/locki ) then
    mv $stanenv/locki $stanenv/lockb
    touch $stanenv/nolocki
  endif
```

```
echo ""
echo ""
# Saisie des informations necessaires
# * Nom de la matrice de demande
echo "2" > $repbat/mfdir
echo "mf" >> $repbat/mfdir
echo "q" >> $repbat/mfdir
( $stammod/s314 < $repbat/mfdir ) >& /dev/null
echo "Enter demand matrix:"
echo -n "      matrix="
set REP=$<
set FLAG=1
while ( $FLAG )
  if ( $REP == "" ) then
    mv $stanenv/rapport $stanenv/reports
    if ( -e $stanenv/nolocki) then
      rm $stanenv/nolocki
      mv $stanenv/lockb $stanenv/locki
    endif
    rm $repbat/*
    rmdir $repbat
    exit
  endif
  grep -s "^ $REP" $stanenv/reports
  if ( $status == 0 ) break
  echo "$REP is not a valid matrix"
  echo -n "   matrix="
  set REP=$<
end
rm $stanenv/reports
set MATOD=$REP
# * Nom de la matrice contenant les totaux aux origines
echo "2" > $repbat/modir
echo "mo" >> $repbat/modir
echo "q" >> $repbat/modir
( $stammod/s314 < $repbat/modir ) >& /dev/null
echo "Enter supply at origin matrix (mo):"
echo -n "      matrix="
set REP=$<
set FLAG=1
while ( $FLAG )
  if ( $REP == "" ) then
    mv $stanenv/rapport $stanenv/reports
    if ( -e $stanenv/nolocki) then
      rm $stanenv/nolocki
      mv $stanenv/lockb $stanenv/locki
    endif
    rm $repbat/*
    rmdir $repbat
    exit
  endif
  grep -s "^ $REP" $stanenv/reports
  if ( $status == 0 ) break
  echo "$REP is not a valid matrix"
  echo -n "   matrix="
  set REP=$<
end
rm $stanenv/reports
set MATO=$REP
# * Nom de la matrice contenant les totaux aux destinations
echo "2" > $repbat/mddir
echo "md" >> $repbat/mddir
echo "q" >> $repbat/mddir
( $stammod/s314 < $repbat/mddir ) >& /dev/null
echo "Enter demand at destinations matrix (md):"
echo -n "      matrix="
set REP=$<
set FLAG=1
while ( $FLAG )
  if ( $REP == "" ) then
    mv $stanenv/rapport $stanenv/reports
    if ( -e $stanenv/nolocki) then
      rm $stanenv/nolocki
      mv $stanenv/lockb $stanenv/locki
    endif
    rm $repbat/*
    rmdir $repbat
    exit
  endif
  grep -s "^ $REP" $stanenv/reports
  if ( $status == 0 ) break
  echo "$REP is not a valid matrix"
  echo -n "   matrix="
```

```
    set REP=$<
  end
  rm $stanenv/reports
  set MATD=$REP

# Preparation des fichiers de dialogue pour STAN
# $repbat/cpcc <--- pour extraire dans la matrice de demande la valeur
#      des plus courts chemins
  echo "2" > $repbat/cpcc
  echo "e" >> $repbat/cpcc
  echo "$MATOD" >> $repbat/cpcc
  echo "n" >> $repbat/cpcc
  echo "q" >> $repbat/cpcc
# $repbat/invm <--- pour inverser la matrice des plus courts chemins
  echo "1" > $repbat/invm
  echo "y" >> $repbat/invm
  echo "$MATOD" >> $repbat/invm
  echo "n" >> $repbat/invm
  echo "(1/$MATOD)" >> $repbat/invm
  echo "" >> $repbat/invm
  echo "$MATOD" >> $repbat/invm
  echo "" >> $repbat/invm
  echo "n" >> $repbat/invm
  echo "q" >> $repbat/invm
# $repbat/balm <--- pour balancer la matrice de demande
  echo "1" > $repbat/balm
  echo "$MATOD" >> $repbat/balm
  echo "$MATO" >> $repbat/balm
  echo "$MATD" >> $repbat/balm
  echo "n" >> $repbat/balm
  echo "" >> $repbat/balm
  echo "1" >> $repbat/balm
  echo "$MATOD" >> $repbat/balm
  echo "n" >> $repbat/balm
  echo "q" >> $repbat/balm

# Calcul de la matrice des plus courts chemins en termes des distances
  ( $stanmod/s233 < $repbat/cpcc ) >& /dev/null

# Inversion de la matrice
  ( $stanmod/s325 < $repbat/invm ) >& /dev/null

# Balancement de  la matrice en fonction des totaux
  ( $stanmod/s326 < $repbat/balm ) >& /dev/null
  rm $stanenv/reports

# Remise des environnements en ordre et nettoyage des fichiers temporaires
  mv $stanenv/rapport $stanenv/reports
  rm $repbat/*
  rmdir $repbat
  if ( -e $stanenv/nolocki ) then
    rm $stanenv/nolocki
    mv $stanenv/lockb $stanenv/locki
  endif

$at eof
```

3. Liste du sous-programme d'exécution sm2

```
$!/bin/csh -f
$*****************************************************************************
# Programme pour simuler, grace a une suite d'appels adequats le deroulement
# de l'algorithme de Frank et Wolfe applique au probleme de trouver
# simultanement les flots optimaux et la matrice de demande quand l'objectif
# ne possede pas de composante economique.
#
# Note importante:
# ----------------
#   --> L'usager peut aussi abandonner l'execution du programme en repondant
#       par une ligne vide a la place d'un nom de matrice, ou par "r" ou "q"
#       au choix du mode de sortie.
#
```

```
$*** Variables globales heritees de la procedure s551:
$    ------------------------------------------------
$        stanenv,stanmod,
$        modenv,repbat    : Repertoires definissant les environnements de travail
$        status           : Octet de statut a la fin d'une execution de programme
$
$*** Variables locales de controle de validite:
$    ------------------------------------------
$        STATUT : Octet de statut du scenario courant
$        ready  : Bit vrai si le scenario est pret pour une affectation
$        PREPOK : Bit vrai si la preparation tient compte des limitations
$                 du programme
$
$*** Variables locales parametres de l'affectation:
$    ----------------------------------------------
$        MATO   : Vecteur colonne des totaux disponibles aux origines
$        MATD   : Vecteur ligne des totaux demandes aux destinations
$        MATOD  : Matrice de demande de l'affectation
$        MATC   : Matrice de travail pour conserver des resultats intermediaires
$                 >Les couts marginaux entre paires o/d apres une affectation
$                 >L'ancienne matrice de demande durant l'affectation de celle
$                  resultant de l'execution de RNET sur le reseau
$        MODES  : Modes de transport utilises pour faire l'affectation
$        PRODS  : Produit affecte
$        IOUT   : Mode selon lequel l'usager desire voir afficher les resultats
$        GAPOK  : Bit vrai si l'usager desire l'utilisation du critere du GAP
$
$*** Variables locales de controle de l'optimalite:
$    ----------------------------------------------
$        ITER   : Numero de la prochaine iteration du module d'affectation
$        MITER  : Critere d'arret sur le nombre d'iterations --- seuil considere
$        GAP    : Critere d'arret sur le GAP --- seuil considere
$        LAMBDA : Valeur du pas optimal
$        OPT    : Bit vrai si l'optimalite a ete atteinte
$
$*** Variables locales de travail:
$    -----------------------------
$        SCENC,scext,ADDSC,RECORD,FLAG,REP,NEWGAP,PR
$*****************************************************************************

# Initialisations
  set ready=0
  set OPT=0

# Statut de traitement par lots forces pour les appels subsequents
  if ( -e $stanenv/locki ) then
    mv $stanenv/locki $stanenv/lockb
    touch $stanenv/nolocki
  endif

# timing...
  if ( -e $stanenv/time ) then
    echo "" >> $stanenv/time
    echo "" >> $stanenv/time
  else
    touch $stanenv/time
  endif
  echo "MODULE 5.51: FRANK & WOLFE ASSIGNMENT" >> $stanenv/time
  echo "------------------------------------" >> $stanenv/time
  echo "" >> $stanenv/time
  echo "beginning of assignment   --> `date +'%D at %T'`"  >> $stanenv/time

# Preparation de l'environnement de travail
  mv $stanenv/reports $stanenv/rapport
  if ( -d $repbat ) then
    rm $repbat/*
  else
    mkdir $repbat
  endif
  if ( -e $repbat/rapport ) rm $repbat/rapport
  cd $stanenv

$*****************************************************************************
# Verifications... de la validite de la requete d'execution
$*****************************************************************************

# Preparation du fichier dialogue de STAN et
# recuperation du numero du scenario courant
  echo "1" > $repbat/stat
  echo "1,1,35,35" >> $repbat/stat
  echo "q" >> $repbat/stat
  ( $stanmod/s111 < $repbat/stat ) >& /dev/null
  set SCENC=`grep '35:' $stanenv/reports | awk -F\: '{print $2}'`

# Preparation du fichier dialogue pour STAN et
# recuperation du numero externe du scenario courant, son statut,
```

```
# de l'iteration actuelle, et du nombre maximal d'iterations a faire
  set ADDSC=0
  @ ADDSC = $ADDSC + $SCENC + 70                    # adresse du numero externe
  echo "1" > $repbat/stat
  echo "1,1,$ADDSC,$ADDSC" >> $repbat/stat
  set RECORD = 0
  @ RECORD = $SCENC + 1                             # adresse de l'enregistrement du scenario
  set ADDSC=7                                       # adresse de l'octet de statut
  echo "1" >> $repbat/stat
  echo "1,$RECORD,$ADDSC,$ADDSC" >> $repbat/stat
  set ADDSC=10                                      # adresse de ITER
  echo "1" >> $repbat/stat
  echo "1,$RECORD,$ADDSC,$ADDSC" >> $repbat/stat
  set ADDSC=11                                      # adresse de NMAXITER
  echo "1" >> $repbat/stat
  echo "1,$RECORD,$ADDSC,$ADDSC" >> $repbat/stat
  echo "q" >> $repbat/stat
  ( $stanmod/s111 < $repbat/stat ) >& /dev/null

# Extraction de l'information recuperee
  set scext='grep "$SCENC\:" $stanenv/reports | awk -F\: '{print $2}''
  set STATUT='grep '7:' $stanenv/reports | awk -F\: '{print $2}''
  @ ready = (( $STATUT >> 3) & 1 )                  # affectation preparee
  set ITER='grep '10:' $stanenv/reports | awk -F\: '{print $2}''
  set NITER='grep '11:' $stanenv/reports | awk -F\: '{print $2}''
  rm $stanenv/reports

# Fin prematuree si l'affectation d'a pas ete preparee
  if ( ! $ready || ( $NITER != 0 && $ITER > $NITER )) then
    echo ""
    echo "^G  Scenario $scext not ready for assignment"
    mv $stanenv/rapport $stanenv/reports
    if ( -e $stanenv/nolocki ) then
      rm $stanenv/nolocki
      mv $stanenv/lockb $stanenv/lockb
    endif
    rm $repbat/*
    rmdir $repbat
    exit
  endif

# Preparation du fichier de dialogue pour STAN et
# saisie: * de la matrice des des totaux de l'offre aux origines (mo)
  echo "2" > $repbat/modir
  echo "mo" >> $repbat/modir
  echo "q" >> $repbat/modir
  ( $stanmod/s314 < $repbat/modir ) >& /dev/null
  echo "Enter supply at origin matrix (mo):"
  echo -n "     matrix="
  set REP=$<
  set FLAG=1
  while ( $FLAG )
    if ( $REP == "" ) then
      mv $stanenv/rapport $stanenv/reports
      if ( -e $stanenv/nolocki) then
        rm $stanenv/nolocki
        mv $stanenv/lockb $stanenv/locki
      endif
      rm $repbat/*
      rmdir $repbat
      exit
    endif
    grep -s "^ $REP" $stanenv/reports
    if ( $status == 0 ) break
    echo "$REP is not a valid matrix"
    echo -n "   matrix="
    set REP=$<
  end
  rm $stanenv/reports
  set MATO=$REP
# saisie: * de la matrice des des totaux de la demande aux origines (md)
  echo "2" > $repbat/mddir
  echo "md" >> $repbat/mddir
  echo "q" >> $repbat/mddir
  ( $stanmod/s314 < $repbat/mddir ) >& /dev/null
  echo "Enter demand at destinations matrix (md):"
  echo -n "     matrix="
  set REP=$<
  set FLAG=1
  while ( $FLAG )
    if ( $REP == "" ) then
      mv $stanenv/rapport $stanenv/reports
```

```
        if ( -e $stanenv/nolocki) then
          rm $stanenv/nolocki
          mv $stanenv/lockb $stanenv/locki
        endif
        rm $repbat/*
        rmdir $repbat
        exit
      endif
      grep -s "" $REP" $stanenv/reports
      if ( $status == 0 ) break
      echo "$REP is not a valid matrix"
      echo -n "   matrix="
      set REP=$<
    end
    rm $stanenv/reports
    set MATD=$REP
  @ saisie: * de la matrice de travail
    echo "2" > $repbat/mfdir
    echo "mf" >> $repbat/mfdir
    echo "q" >> $repbat/mfdir
    ( $stanmod/s314 < $repbat/mfdir ) >& /dev/null
    echo "Enter working matrix:"
    echo -n " matrix="
    set REP=$<
    set FLAG=1
    while ( $FLAG )
      if ( $REP == "" ) then
        mv $stanenv/rapport $stanenv/reports
        if ( -e $stanenv/nolocki) then
          rm $stanenv/nolocki
          mv $stanenv/lockb $stanenv/locki
        endif
        rm $repbat/*
        rmdir $repbat
        exit
      endif
      grep -s "" $REP" $stanenv/reports
      if ( $status == 0 ) break
      echo "$REP is not a valid matrix"
      echo -n " matrix="
      set REP=$<
    end
    rm $stanenv/reports
    set MATC=$REP

  @ Verification du respect des restrictions sur l'affectation:
  @ Preparation du fichier de dialogue pour STAN pour analyser le rapport
  @ le resume de l'affectation qui va etre effectuee
    echo "1" > $repbat/laff
    echo "q" >> $repbat/laff
    ( $stanmod/s601 < $repbat/laff ) >& /dev/null
    set PREPOK='grep " mf" $stanenv/reports | awk 'END {if (NR < 2) print "1" ; else print "0"}''
    if ( ! $PREPOK ) then
      echo ""
      echo ""G  Scenario $ascrt not properly prepared for module 5.51"
      mv $stanenv/rapport $stanenv/reports
      if ( -e $stanenv/nolocki ) then
        rm $stanenv/nolocki
        mv $stanenv/lockb $stanenv/locki
      endif
      rm $repbat/*
      rmdir $repbat
      exit
    endif
  @ Saisie des parametres de l'affectation : le produit affecte, la matrice de
  @ demande, les modes de transport consideres
    echo "6" > $repbat/lprod
    echo "y" >> $repbat/lprod
    echo "q" >> $repbat/lprod
    ( $stanmod/s202  < $repbat/lprod ) >& /dev/null
    set PR='grep " mf" $stanenv/reports | awk '{for (i=1;i<=NF-2;i++) printf $i" "; print "" }''
    set PRODS='grep "p=" $stanenv/d202.out | grep "$PR" | tr "=" " " | awk '{print $2}''
    set MATOD='grep " mf" $stanenv/reports | awk '{print $(NF-1)}''
    set MODES='grep " mf" $stanenv/reports | awk '{print $(NF)}''
    rm $stanenv/d202.out

  @------------------------------------------------------------------------
  @ FIN des verifications...
  @------------------------------------------------------------------------

  @ Selection par l'usager du mode de sortie des resultats.
  @ L'usager a aussi l'opportunite d'abandonner la procedure
```

```
    if ( ! -e $stanenv/nolocki ) then
      set IOUT=2
    else
      echo ""
      echo " Select: List device"
      echo "         1=Terminal"
      echo "         2=Printer"
      set REP=$<
      set FLAG=1
        while ( $FLAG )
          switch ( $REP )
            case [1,2] :
              set FLAG=0
              set IOUT=$REP
              breaksw
            case [r] :
              echo "*** ABORTED ***    date +%T" >> $stanenv/time
              mv $stanenv/rapport $stanenv/reports
              if ( -e $stanenv/nolocki ) then
                rm $stanenv/nolocki
                mv $stanenv/lockb $stanenv/locki
              endif
              rm $repbat/*
              rmdir $repbat
              exit
              breaksw
            case [q] :
              echo "*** ABORTED ***    date +%T" >> $stanenv/time
              mv $stanenv/rapport $stanenv/reports
              if ( -e $stanenv/nolocki ) then
                rm $stanenv/nolocki
                mv $stanenv/lockb $stanenv/locki
              endif
              rm $repbat/*
              rmdir $repbat
              exit 2
              breaksw
            default :
              echo "valid answer are 1, 2, r for restart and q for quit"
              set REP=$<
              breaksw
          endsw
      end
    endif

  # timing...
    echo "End of verifications   --> `date +%T`" >> $stanenv/time
    echo "" >> $stanenv/time

    touch $repbat/rapport

  # Recuperation des numeros d'origines et de destinations
  # et modification des parametres du modules de sortie
    echo "4" > $repbat/pmatr
    echo "2" >> $repbat/pmatr
    echo "8,0" >> $repbat/pmatr
    echo "4" >> $repbat/pmatr
    echo "3" >> $repbat/pmatr
    echo "$MATO" >> $repbat/pmatr
    echo "" >> $repbat/pmatr
    echo "n" >> $repbat/pmatr
    echo "n" >> $repbat/pmatr
    echo "q" >> $repbat/pmatr
    ( $stammod/s314 < $repbat/pmatr ) >& /dev/null
  # Premiere origine
    set O1=`tail +31 $stanenv/d314.out|head -1|awk '{print $1}'`
  # Derniere origine
    set Oo=`tail -21 $stanenv/d314.out|head -1|awk '{print $1}'`
    echo "4" > $repbat/pmatr
    echo "2" >> $repbat/pmatr
    echo "8,0" >> $repbat/pmatr
    echo "4" >> $repbat/pmatr
    echo "3" >> $repbat/pmatr
    echo "$MATD" >> $repbat/pmatr
    echo "" >> $repbat/pmatr
    echo "n" >> $repbat/pmatr
    echo "n" >> $repbat/pmatr
    echo "q" >> $repbat/pmatr
    ( $stammod/s314 < $repbat/pmatr ) >& /dev/null
  # Premiere destination
    set D1=`tail +31 $stanenv/d314.out|head -1|tr ":" " "|awk '{print $2}'`
  # Derniere destination
```

```
  set Dd=`tail -21 $stanenv/d314.out|head -1|tr ":" " "|awk '{print $(NF-1)}'`

# Initialisation: Creation de tous les fichiers de dialogue fixes
# (i.e. qui ne changeront pas au cours de l'execution du programme)
# pour les appels subsequents au modules de STAN
#
# $repbat/pmatr <--- pour extraire dans le fichier $stanenv/d314.out
#                    la matrice des couts marginaux qui sera lu par
#                    l'interface RNET
  echo "4" > $repbat/pmatr
  echo "2" >> $repbat/pmatr
  echo "8,0" >> $repbat/pmatr
  echo "4" >> $repbat/pmatr
  echo "3" >> $repbat/pmatr
  echo "$MATO" >> $repbat/pmatr
  echo "" >> $repbat/pmatr
  echo "n" >> $repbat/pmatr
  echo "n" >> $repbat/pmatr
  echo "3" >> $repbat/pmatr
  echo "$MATD" >> $repbat/pmatr
  echo "y" >> $repbat/pmatr
  echo "n" >> $repbat/pmatr
  echo "n" >> $repbat/pmatr
  echo "3" >> $repbat/pmatr
  echo "$MATC" >> $repbat/pmatr
  echo "y" >> $repbat/pmatr
  echo "n" >> $repbat/pmatr
  echo "n" >> $repbat/pmatr
  echo "y" >> $repbat/pmatr
  echo "$O1,$Oo" >> $repbat/pmatr
  echo "" >> $repbat/pmatr
  echo "$D1,$Dd" >> $repbat/pmatr
  echo "" >> $repbat/pmatr
  echo "q" >> $repbat/pmatr
# $repbat/cpmat <--- pour copier sauvegarder la matrice de demande
#                    dans celle de travail
  echo "4" > $repbat/cpmat
  echo "1" >> $repbat/cpmat
  echo "$MATOD" >> $repbat/cpmat
  echo "$MATC" >> $repbat/cpmat
  echo "n" >> $repbat/cpmat
  echo "n" >> $repbat/cpmat
  echo "q" >> $repbat/cpmat
# $repbat/emat  <--- entete du fichier de modification interactive
#                    de la matrice de demande - le reste du fichier
#                    est cree par l'interface RNET
  echo "3" > $repbat/emat
  echo "$MATOD" >> $repbat/emat
  echo "1" >> $repbat/emat
  echo "$O1,$D1" >> $repbat/emat
# $repbat/lmatr <--- pour ecrire dans le rapport les valeurs de la
#                    matrice O/D actuelle
  echo "1" > $repbat/lmatr
  echo "$MATOD" >> $repbat/lmatr
  echo "" >> $repbat/lmatr
  echo "1" >> $repbat/lmatr
  echo "n" >> $repbat/lmatr
  echo "n" >> $repbat/lmatr
  echo "1" >> $repbat/lmatr
  echo "y" >> $repbat/lmatr
  echo "$O1,$Oo" >> $repbat/lmatr
  echo "" >> $repbat/lmatr
  echo "$D1,$Dd" >> $repbat/lmatr
  echo "" >> $repbat/lmatr
  echo "q" >> $repbat/lmatr
# $repbat/xcmrg <--- pour sauvegarder dans la matrice MATC la valeur
#                    des couts marginaux sur les chemins les plus
#                    courts
  echo "2" > $repbat/xcmrg
  echo "3" >> $repbat/xcmrg
  echo "$MODES" >> $repbat/xcmrg
  echo "$PRODS" >> $repbat/xcmrg
  echo "" >> $repbat/xcmrg
  echo "$MATC" >> $repbat/xcmrg
  echo "n" >> $repbat/xcmrg
  echo "q" >> $repbat/xcmrg
```

```
# $repbat/envb <--- pour sauvegarder le "path" et le nom du fichier
#                   d'entree pour rnet, les numeros des premier et dernier
#                   centroides d'offre et de demande
  echo "$O1" > $repbat/envb
  echo "$Oo" >> $repbat/envb
  echo "$D1" >> $repbat/envb
  echo "$Dd" >> $repbat/envb
  echo "$stanenv/d314.out" >> $repbat/envb

# Rapport du resume de l'affectation et analyse pour detecter
# le gap a ete demande comme critere d'arret
  sed -e 's/Module: 6.01/Module: 5.51/g' $stanenv/reports > $repbat/rapport
  grep -s 'gap:' $stanenv/reports
  if ( $status == 1) then
    set GAPOK=0
  else
    set GAPOK=1
    set GAP="grep 'gap:' $stanenv/reports | tr ':.%' ' ' | awk '{print ($3,$4)}'"
  endif
  rm $stanenv/reports

# Preparation du fichier de dialogue pour STAN pour n'effectuer qu'une
# seule iteration avec le module 5.21 : MMATXIT <--- ITER
  set ADDSC=11
  echo "3" > $repbat/mstat
  echo "1,$RECORD,$ADDSC,$ADDSC" >> $repbat/mstat
  echo "$ITER" >> $repbat/mstat
  echo "q" >> $repbat/mstat
  ( $stanmod/s111 < $repbat/mstat ) >& /dev/null

#-------------------------------------------------------#
# Iteration initiale si l'affectation est nouvelle #
#-------------------------------------------------------#

  if ( $ITER == 0 ) then
#   timing...
    echo "Beginning of iteration 0 at 'date +%T'" >> $stanenv/time
    echo "=================================" >> $stanenv/time
    ( $stanmod/s521 ) >& /dev/null
    tail +261 $stanenv/reports >> $repbat/rapport
    grep -s 'L\*' $stanenv/reports
    if ( status == 1 && $GAPOK ) then
      set NEWGAP="grep 'Gap:' $stanenv/reports | tr ':.%' ' ' | awk '{print ($2,$3)}'"
      if ( $NEWGAP[1] < $GAP[1] ) then
        set OPT=1
      else
        if ( $NEWGAP[1] == $GAP[1] && $NEWGAP[2] <= $GAP[2] ) then
          set OPT=1
        endif
      endif
    endif
    rm $stanenv/reports
    @ ITER = $ITER + 1
  endif

# Impression de la matrice O/D dans le rapport d'iterations
    ( $stanmod/s314 < $repbat/lmatr ) >& /dev/null
    sed -e 's/Module: 3.14/Module: 5.51/g' $stanenv/reports >> $repbat/rapport
    rm $stanenv/reports

#   Redaction du rapport
#   timing...
    echo "Print reports           -->   'date +%T'" >> $stanenv/time
    echo "" >> $stanenv/time
    if ( $IOUT == 1 ) then
      more $repbat/rapport
    else
      cat $repbat/rapport >> $stanenv/rapport
    endif
    rm $repbat/rapport
    touch $repbat/rapport

#*************************************************************************************
# Boucle principale de calcul
#*************************************************************************************
    while ( $ITER <= $NITER && ! $OPT )

# timing...
      echo "Beginning of iteration $ITER at 'date +%T'" >> $stanenv/time
      if ( $ITER <= 9 ) then
        echo "=================================" >> $stanenv/time
      else
        echo "==================================" >> $stanenv/time
```

```
      endif

# Recuperation des couts marginaux de l'affectation precedente
# timing...
      echo "Marginal costs extr   --> `date +%T`" >> $stanenv/time
      ( $stanmod/s633 < $repbat/xcmrg ) >& /dev/null
      ( $stanmod/s314 < $repbat/pmatr ) >& /dev/null

# Appel au programme d'interface avec la procedure RNET
# timing...
      echo "Calling RNET   --> `date +%T`" >> $stanenv/time
      cd $modenv
      rnet >& /dev/null
      cd $stanenv

# Sauvegarde de la matrice O/D actuelle pour l'etape de mise a jour ulterieure
# timing...
      echo "Saving result   --> `date +%T`" >> $stanenv/time
      ( $stanmod/s312 < $repbat/cpmat ) >& /dev/null

# Recuperation de la matrice O/D resultat de l'execution de RNET
# Le fichier mmat1 a ete mis a jour par le programme rnet
# La partie emat est fixe et a ete preparee au debut du programme
      cat $repbat/emat $repbat/mmat1 > $repbat/mmat
      echo "q" >> $repbat/mmat
      ( $stanmod/s312 < $repbat/mmat ) >& /dev/null

# Preparation du fichier de dialogue pour STAN et
# preparation de la banque pour une iteration supplementaire du
# module d'affectation, puis execution de l'iteration
# timing...
      echo "Assigning new O/D matrix  --> `date +%T`" >> $stanenv/time
      echo "1" > $repbat/saff
      echo "1" >> $repbat/saff
      echo "$ITER" >> $repbat/saff
      if ( $GAPOK ) then
        echo "y" >> $repbat/saff
        echo "$GAP[1].$GAP[2]" >> $repbat/saff
      else
        echo "n" >> $repbat/saff
      endif
      ( $stanmod/s511 < $repbat/saff ) >& /dev/null
      ( $stanmod/s521 ) >& /dev/null
      head -5 $stanenv/reports | sed -e 's/Module: 5.21/Module: 5.51/g' >> \
$repbat/rapport
      if ( $GAPOK ) then
        set NEWGAP=`grep 'Gap:' $stanenv/reports | tr ':.%' ' ' ' ' | awk '{print ($2,$3)}'`
        if ( $NEWGAP[1] < $GAP[1] ) then
          set OPT=1
        else
          if ( $NEWGAP[1] == $GAP[1] && $NEWGAP[2] <= $GAP[2] ) then
            set OPT=1
          endif
        endif
      endif
      tail +261 $stanenv/reports >> $repbat/rapport

# Regarder le statut de la recherche lineaire dans le scenario
# Si elle a lieu (ie : cout non-lineaires) recuperation de la valeur de Lambda
# Sinon, poser Lambda=1 et marquer l'optimalite
      echo "Line search value   --> `date +%T`" >> $stanenv/time
      grep -s 'L\*' $stanenv/reports
      if ( $status == 1 ) then
          set OPT=1
          set LAMBDA=1
      else
        set LAMBDA=`grep 'L\*' $stanenv/reports | awk -F: '{print $2}'`
      endif
      rm $stanenv/reports

# Preparation du fichier de dialogue pour STAN et
# mise a jour de la matrice de demande a savoir :
# MATOD = (1-LAMBDA)*MATC + LAMBDA*MATOD
# timing...
      echo "O/D matrix adjustment   --> `date +%T`" >> $stanenv/time
      echo "1" > $repbat/cmatr2
      echo "y" >> $repbat/cmatr2
      echo "$MATOD" >> $repbat/cmatr2
      echo "n" >> $repbat/cmatr2
      echo "(((1-$LAMBDA)*$MATC)+($LAMBDA*$MATOD))" >> $repbat/cmatr2
      echo "" >> $repbat/cmatr2
      echo "" >> $repbat/cmatr2
```

```
      echo "y" >> $repbat/cmatr2
      echo "$01,$0o" >> $repbat/cmatr2
      echo "" >> $repbat/cmatr2
      echo "$D1,$Dd" >> $repbat/cmatr2
      echo "" >> $repbat/cmatr2
      echo "q" >> $repbat/cmatr2
      ( $stanmod/s325 < $repbat/cmatr2 ) >& /dev/null
      rm $stanenv/reports

# Impression de la matrice O/D dans le rapport d'iterations
      ( $stanmod/s314 < $repbat/lmatr ) >& /dev/null
      sed -e 's/Module: 3.14/Module: 5.51/g' $stanenv/reports >> $repbat/rapport
      rm $stanenv/reports

# Redaction du rapport
# timing...
      echo "Print reports   --> `date +%T`" >> $stanenv/time
      echo "" >> $stanenv/time
      if ( $IOUT == 1 ) then
        more $repbat/rapport
      else
        cat $repbat/rapport >> $stanenv/rapport
      endif
      rm $repbat/rapport
      touch $repbat/rapport

# Incrementer le compteur d'iteration
      @ ITER = $ITER + 1

    end

#*****************************************************************************
# FIN de la boucle principale
#*****************************************************************************

# Remise des environnements en ordre et nettoyage des fichiers temporaires
  mv $stanenv/rapport $stanenv/reports
  rm $repbat/*
  rmdir $repbat
  if ( -e $stanenv/nolocki ) then
    rm $stanenv/nolocki
    mv $stanenv/lockb $stanenv/locki
  endif

$at eof
```

4. Liste de l'interface avec RNET

```
      program interface
C*****************************************************************************
C* Cette procedure sert d'interface entre le module s551 de STAN et
C* la librairie RNET
C*
C* Dans une premiere etape, elle lit dans le fichier
C* '/usr/crts/christin/finland/finbank/bin/s551.sm/tmp' de l'environnement
C* du module les numeros du premier et du dernier centroide d'offre, du
C* premier et du dernier centroide de demande. Cette information permet de
C* connaitre immediatement le format du fichier d'entree et donc de le
C* lire d'une seul trait sans etre oblige de proceder a des conversions de
C* type en cours de routes. De plus, elle sera utile au moment de preparer
C* le fichier de modification de la matrice demande qui sera utilise par
C* STAN. Enfin, elle lira le nom et l'adresse du fichier de donnees.
C*
C* Puis, elle ouvre le fichier de donnees et lit l'offre, la demande,
C* et les couts des arcs du reseau.
C*
C* Une fois toutes ces informations saisies, le reseau interne est construit
C* selon le format de RNET, puis cette derniere procedure est appelee.
C* Au retour de l'appel, les resultats, c'est-a-dire les flots a moindre
C* couts sur le reseau de transport sont extraits et sauvegardes dans la
C* nouvelle matrice de demande.
C*
C* Enfin, les intructions pour creer cette nouvelle matrice sont consignees
C* dans un fichier de l'environnement temporaire du module dans le format
C* necessaire pour etre interpretees par STAN.
C*
```

ANNEXE B Introduction à STAN

Cette annexe est divisée en trois sections. Dans la première, nous décrirons succinctement la banque de données et son environnement. Dans la seconde, nous donnerons une brève description des modules disponibles dans la version actuelle de STAN. Enfin, dans la troisième, le lecteur trouvera un complement bibliographique concernant STAN. Ainsi pourrat-il consulter [26] pour un survol du projet dans lequel STAN a vu le jour, [27] pour les détails de la modélisation et des algorithmes utilisés par STAN, [28] pour une introduction complète, mais moins détaillée que [3], à l'usage de STAN, et enfin [29] et [30] pour la description détaillée d'une application.

1. Environnement de la banque

La banque de données STAN (*stanbank*) est un gros fichier permanent dans lequel sont consignées les données relatives à une, ou plusieurs versions d'un réseau de transport. Chacune de ces versions est appelée scénario. Les scénarios peuvent différer aussi bien en termes du réseau de transport (nœuds et arcs), qu'en termes des modes définis ou des produits transportés.

Autour de *stanbank* gravitent six autres fichiers, dont trois sont réellement utiles à l'usager, les trois autres étant utilisés par STAN.

Les trois premiers sont:

errors, fichier où est consigné le code de toute erreur détectée en cours d'exécution, ainsi que le numéro du module où elle a eu lieu et les paramètres en jeu (voir [3] pour la liste des codes d'erreur).

reports, fichier contenant les visualisations sous forme de listes et les rapports d'exécution demandés par l'usager.

plots, fichier contenant les graphiques générés par STAN à la demande de d'usager.

Les trois autres sont *nodata*, *nextmod*, et *lock*[*i*|*b*]. Mentionnons seulement à propos du dernier qu'il est crée lors de l'appel initial à STAN, et que sa présence interdit à tout autre usager d'accéder concurremment à la banque de données. De plus, *locki* indique un appel en mode interactif, alors que *lockb* indique un appel en mode de traitement par lots.

2. Les modules et leurs fonctions

Le progiciel STAN est un ensemble de programmes indépendants, appelés modules, qui accèdent à une librairie commune de procédures. Les modules sont regroupés en six ensembles:

Groupe 1: Utilitaires.

Il s'agit d'un ensemble de procédures permettant de manipuler directement le fichier *stanbank* ; de consulter le registre des personnes ayant accédé à la banque de données récemment; de créer, supprimer, copier ou décider du niveau de protection des scénarios de la banque; et enfin, de créer, supprimer ou modifier certaines données utiles, mais non essentielles, de la banque comme les titres de projets et de scénarios ou les lignes de démarcations pour le réalisme du graphisme.

```
1.   U T I L I T I E S
1.11 Direct data base manipulations
1.21 Log book consultation
1.22 Scenario manipulations
1.23 Input / modify / output utility data
1.31 Input          demarcation lines using batch entry
1.32 Input / modify demarcation lines interactively
1.34 Output         demarcation lines

2.   N E T W O R K   E D I T O R
2.01 Input / modif  / output modes
2.02 Input / modif  / output products
2.11 Input          base network using batch entry
2.12 Input / modify base network interactively
2.13 Plot           base network
2.14 Output         base network
2.21 Input          transfers using batch entry
2.22 Input / modify transfers interactively
2.23 Plot           transfers
2.24 Output         transfers
2.33 Plot shortest paths on network

3.   M A T R I X   E D I T O R
3.01 Input / modify / display zone groups
3.11 Input          matrices using batch entry
3.12 Input / modify matrices interactively
3.13 Plot           matrices
3.14 Output         matrices
3.15 Plot           matrix histograms
3.25 Matrix calculations
3.26 Matrix balancing

4.   F U N C T I O N   E D I T O R
4.01 Input / modify / display function classes
4.11 Input          functions using batch entry
4.12 Input / modify functions interactively
4.13 Plot           functions
4.14 Output         functions

5.   A S S I G N M E N T   P R O C E D U R E S
5.11 Prepare scenario for   assignment
5.21 Multiproduct multimode assignment
5.31 Maxflow calculations

6.   R E S U L T S
6.01 Result summary
6.13 Plot    product costs and volumes
6.14 List    product results
6.15 Plot    product vehicles and convoys
6.23 Compare product costs and volumes
6.33 Plot    shortest paths on network

9.   E N D   O F   S E S S I O N
```

Figure B1 Modules actuellement opérationnels dans STAN.

Groupe 2: Editeur de réseau.

Cet ensemble de modules permet de créer, modifier et visualiser graphiquement ou sous forme de listes les diverses composantes du réseau du scénario courant, à savoir: les modes disponibles, les produits, les nœuds, les liens et les transferts. Il permet également de calculer et visualiser les chemins les plus courts en termes des distances réelles dans le réseau.

Groupe 3: Editeur de matrices.

Cet ensemble permet de créer, modifier, visualiser graphiquement, ou sous forme de listes ou d'histogrammes les matrices[5] de la banque de données. Il permet également de créer, modifier et visualiser sous forme de listes des regroupements de centroïdes. Enfin, le groupe comprend une calculatrice à matrices et des procédures de balancement.

Groupe 4: Editeur de fonctions.

Il permet de créer, supprimer, modifier et visualiser les classes[6] de fonctions et les fonctions de coût.

Groupe 5: Procédures d'affectation.

Ces procédures permettent de préparer une affectation pour le scénario courant, c'est-à-dire de spécifier quels produits affecter, sur quels ensembles de modes chacun et avec quelles matrices de demande. Elles permettent également, d'effectuer une AMF. C'est dans ce groupe que peut s'insérer le module expérimental **5.51** présenté dans le présent mémoire.

---

[5]Les matrices sont des scalaires (matrices 1 x 1), des matrices de demande pleines et des vecteurs-lignes ou colonnes correspondant à l'offre et la demande respectivement.

[6]Il est possible dans STAN de séparer le coût en quatre composantes: le coût d'opération, le coût dû au délais, le coût dû à la consomation d'énergie et un coût supplémentaire défini par l'usager. Au lieu de donner à chaque lien la fonction de chacun de ces coûts comme attribut, on lui donne un numéro de classe de fonction; et c'est dans la définition de celle-ci que l'on défini les fonctions qui lui sont attribuées. Ceci permet de modifier plus aisément la structure de coûts du réseau de transport.

Groupe 6: Résultats d'affectation.

Ces modules permettent d'obtenir sous forme graphique (ou de liste) divers résultats relatifs à un scénario donné, ou à la comparaison de deux scénarios différents.

3. Bibliographie

[26] *Strategic Planning of Freight Transportation in Brazil: Methodology and Applications — Part 1 — Project Summary*, ed. par Florian, M. et Crainic, T. G., Publication #638, Centre de recherche sur les transports, Université de Montréal, 1989.

[27] *Strategic Planning of Freight Transportation in Brazil: Methodology and Applications — Part 2 — Methodology: The New Contributions*, ed. par Florian, M. et Crainic, T. G., Publication #639, Centre de recherche sur les transports, Université de Montréal, 1989.

[28] *Strategic Planning of Freight Transportation in Brazil: Methodology and Applications — Part 3 — The STAN System*, ed. par Florian, M. et Crainic, T. G., Publication #640, Centre de recherche sur les transports, Université de Montréal, 1989.

[29] *Strategic Planning of Freight Transportation in Brazil: Methodology and Applications — Part 4 — Specifying Data for the STAN System: Requirements, Concepts, Sources and Estimation Procedures*, ed. par Florian, M. et Crainic, T. G., Publication #641, Centre de recherche sur les transports, Université de Montréal, 1989.

[30] *Strategic Planning of Freight Transportation in Brazil: Methodology and Applications — Part 5 — Applications*, ed. par Florian, M. et Crainic, T. G., Publication #642, Centre de recherche sur les transports, Université de Montréal, 1989.

Remerciements

Cette étude n'aurait pas vu le jour sans l'aide, la disponibilité et les précieux conseils de Monsieur Michael Florian, mon directeur. Je tiens à lui exprimer ici toute ma reconnaissance.

Je remercie également Monsieur Teodor Gabriel Crainic, particulièrement pour sa patiente assistance durant mon apprentissage de l'usage de STAN.

Enfin, toute ma gratitude et mon respect va à mes parents, sans les nombreux sacrifices desquels mes études universitaires seraient restées du domaine du rêve.